

USAISEC

*US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300*

(Handwritten signature)

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

AD-A268 578

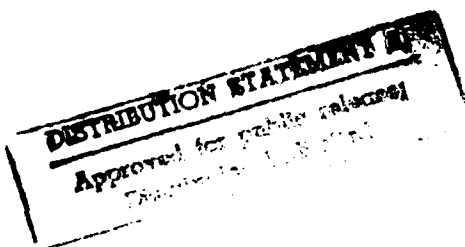


**Army Non-Programmer System
for Working Encyclopedia Request
(ANSWER) - Final Report**

ASQB-GI-92-013

September 1992

**DTIC
SELECTE
AUG 25 1993
S D**



**AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800**



93-19667



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

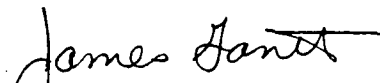
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS NONE	
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT N/A	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A				
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A	
6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)		7a. NAME OF MONITORING ORGANIZATION N/A	
6c. ADDRESS (City, State, and Zip Code)			7b. ADDRESS (City, State, and ZIP Code) N/A	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS	8b. OFFICE SYMBOL (If applicable) ASQB-GCI		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg. Georgia Institute of Technology Atlanta, GA 30332-0800			10. SOURCE OF FUNDING NUMBERS	
			PROGRAM ELEMENT NO.	PROJECT NO.
			TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) Army Nonprogrammer System for Working Encyclopedia Requests, Final Report				
12. PERSONAL AUTHOR(S) Dr. Jim Richardson				
13a. TYPE OF REPORT Technical Paper	13b. TIME COVERED FROM Apr 91 TO Mar 92	14. DATE OF REPORT (Year, Month, Day) July 6, 1992	15. PAGE COUNT 70	
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUBGROUP		
			Heterogeneous Databases, Distributed Processing, SQL Query, Data Management, Schema Integration	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)				
<p>This report details the research efforts into the access of distributed heterogeneous database through an encyclopedia facility. Specifically, several data management tools have been prototyped and are described.</p>				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL LTC David S. Stevens			22b. TELEPHONE (Include Area Code) (404) 894-3110	22c. OFFICE SYMBOL ASQB-GCI

This research was performed by Honeywell Federal Systems, contract number DAKF11-88-24, for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U. S. Army Information Systems Engineering Command (USAISEC). This final report discusses a set of data management tools and distributed query processing modules developed during the contract. Requests to view a demonstration of the prototype may be made by contacting LTC David S. Stevens at 404/894-3110. This research report is not to be construed as an official Army Position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED



Glenn E. Racine, Chief
Computer and Information
Systems Division



James D. Gantt, Ph.D.
Director
AIRMICS

DTIC QUALITY INSPECTED 3

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Spec	Special
A-1	



ANSWER

Final Report

Prepared for:

**AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, Georgia 30332-0800**

Prepared by:

**HFS Incorporated
Federal Systems Operation
7900 Westpark Drive
McLean, Virginia 22102**

and

**Honeywell Incorporated
Sensor and System Development Center
3660 Technology Drive
Minneapolis, Minnesota 55418**

July 6, 1991

Contract Number: DAKF11-88-C-0024

Table of Contents

Section	Page
1	Introduction
1.1	Problem Statement
1.2	ANSWER Program Objectives
1.3	Program Accomplishments
1.3.1	Phase I (May 17, 1988–May 16, 1989)
1.3.2	Phase II (June 16, 1989–December 15, 1989)
1.3.3	Phase IIIA (December 16, 1989–September 30, 1990)
1.3.4	Phase IIIB (October 1, 1990–March 30, 1990)
1.3.5	Phase IV (July 1, 1991–March 31, 1992)
1.4	Current ANSWER Prototype Functions and Architecture
1.5	Standards-Based ANSWER Architecture
1.6	Recommendations for Future Activities
2	Functional Requirements of ANSWER System
2.1	Functional Components
2.2	Detailed Description of Data Models
2.2.1	ECR Model
2.2.2	Relational Model
2.2.3	ECR to Relational Schema Mapping
2.3	Detailed Description of Functional Components
2.3.1	IRDS
2.3.2	Browser
2.3.3	Schema Integrator
2.3.4	Query Formulator
2.3.5	Query Processor
2.3.6	Relational-to-ECR Converter
3	User Interface to ANSWER
3.1	Interface Organizers
3.1.1	Schema Windows
3.1.2	Query Windows
3.1.3	Dialog Boxes
3.1.4	Message Boxes
3.2	Dialog Flow in ANSWER
3.3	Interface Components
3.3.1	Mouse Cursor
3.3.2	Menu Bar
3.3.3	Pushbutton
3.3.4	Node Button
3.3.5	Radio Button

Table of Contents (concluded)

Section		Page
	3.3.6 Textual List	3-4
	3.3.7 Edit Box	3-4
	3.3.8 Text Box	3-4
	3.3.9 System Text	3-4
3.4	Interface Design	3-5
	3.4.1 Main Window	3-5
	3.4.2 Schema Window	3-5
	3.4.3 Query Window	3-9
	3.4.4 Dialog Boxes	3-9
	3.4.5 Message Boxes	3-16
4	ANSWER Installation and Startup	4-1
	4.1 ANSWER System Installation	4-1
	4.2 Starting up Informix	4-2
	4.3 Starting up Oracle	4-2
	4.4 Starting and Exiting ANSWER	4-3
5	References	5-1

List of Figures

Figure		Page
1-1	Engineering and Accessing Databases	1-7
1-2	Architecture of Current ANSWER Prototype	1-8
1-3	Standards-Based ANSWER Architecture	1-9
2-1	IRD Schema for ECR Schema Information	2-4
2-2	OF-AF-RF-DF File Syntax	2-5
2-3	Syntax for SQL Table Definition	2-6
2-4	ANSWER Subset of SQL Data Manipulation Language	2-7
2-5	Schema Integration Script	2-11
2-6	Schema Integrator Architecture	2-11
2-7	Relational-to-ECR Conversion Function	2-13
3-1	Main Window: Schema Menu	3-7
3-2	Main Window: Node Menu	3-8
3-3	Query Window	3-10
3-4	Query Window: Query Menu	3-11
3-5	Query Window: Edit Menu	3-12
3-6	Open Schema Dialog Box	3-17
3-7	Save As Dialog Box	3-18
3-8	Add Node Dialog Box	3-19
3-9	Find Node Dialog Box	3-20
3-10	Open Query Dialog Box	3-21

List of Figures (concluded)

Figure		Page
3-11	Edit Attribute Dialog Box	3-22
3-12	Edit Node Dialog Box	3-23
3-13	Assert Attribute Equivalence Dialog Box	3-24
3-14	Assert E/C Equivalence Dialog Box	3-25
3-15	Delete? Message Box	3-27
3-16	Quit? Message Box	3-28
3-17	Legend Message Box	3-29
3-18	Save? Message Box	3-30

List of Tables

Table		Page
2-1	Correspondence Between Relationship Type and Cardinalities	2-3
3-1	Menus and Menu Items—Main ANSWER Window	3-6
3-2	Query Window Menus	3-9
3-3	Standard ANSWER Dialog Boxes	3-13
3-4	Standard ANSWER Message Boxes	3-26

Section 1

Introduction

This report summarizes the accomplishments of the Army's Nonprogramming System for Working Encyclopedia Requests (ANSWER) program. This multiphase program has been sponsored by the U.S. Army Institute for Research in Management Information, Communications, and Computer Science (AIRMICS). It has been carried out by HFSI (formerly Honeywell Federal Systems Inc.) and Honeywell Sensor and System Development Center between May 1988 and March 1992. The report is intended to be a stand-alone document covering the entire program, but special emphasis is given to Phase IV (the final phase), to the state of the ANSWER system at the end of this phase, and to recommendations for further work. Additional information is contained in the reports of earlier phases [ANSWER88, ANSWER89a, ANSWER89b, ANSWER90, ANSWER91].

This introduction includes:

- Statement of the overall problem that the ANSWER program addresses,
- Program objectives,
- Summary of program accomplishments,
- Brief description of the current ANSWER system,
- Concept for a standards-based ANSWER architecture using commodity software,
- Recommendations for future work.

Section 2 defines the ANSWER system functional requirements (its externally visible behavior) in more detail.

Section 3 describes the ANSWER user interface design, which defines how the user interacts with the prototype. It serves as a user manual for the ANSWER system.

Section 4 discusses installation of the ANSWER software on one or more Sun workstations.

1.1 Problem Statement

The Army, like most large organizations, has developed many information systems to help manage its operations. These information systems exhibit three common forms of heterogeneity:

- Different hardware and operating system platforms,
- Different file systems and database management systems,
- Different kinds of information and structuring of that information.

A user who needs data from more than one information system must know which information systems contain the data, how to access those systems and request the data, and how to combine and correlate the data from different systems to produce an answer to the original query.

The Army is taking steps to facilitate access to information in such a large, heterogeneous and distributed environment by:

- Defining the Army information model,
- Defining an Army information engineering methodology,
- Instituting the Army Data Management and Standards Program [AR 25-9],
- Developing the Army Data Dictionary.

These steps constitute a top-down effort to define a consistent model the Army's operations and to make that model available to application developers. Additional work is needed to tie the Army information model and data dictionary to specific information systems and to provide convenient access to those information systems. This is the focus of the ANSWER program.

1.2 ANSWER Program Objectives

The objective of the ANSWER program is to conduct research and develop prototype tools that provide uniform, integrated access to the Army's metadata and to the information systems themselves. The ANSWER system is a prototype of a comprehensive solution to many of the problems associated with management of large amounts of information physically distributed among many heterogeneous computer systems.

The following text from the program proposal captures the original vision for ANSWER:

Honeywell's multidisciplinary team, in cooperation with the Army, proposes to solve these problems by researching, designing, and prototyping the Army's Nonprogrammer System for Working Encyclopedia Requests (ANSWER). ANSWER is a system that represents a comprehensive solution to many of the problems associated with the management of large amounts of information physically distributed among many heterogeneous computer systems. It consists of the Army Encyclopedia, a unified reference catalog for the processes, information classes, and associated databases of the Army Information Model, and the tools to aid Army personnel in locating and retrieving relevant data from the numerous Army databases and files. Specifically, ANSWER will provide Army personnel with the following capabilities:

- 1. A simple and uniform representation scheme that supports the high-level concepts (information classes and processes) defined in the Army Information Model. This scheme will be adaptable to accommodate changes in the model, and will be capable of representing the security requirements of the model.*
- 2. A browsing capability that allows users to examine the high-level concepts defined in the Army Information Model, and to relate them to the actual data residing in one or more databases and files. This capability allows users who are unfamiliar with the type and/or semantics of the data residing in the databases and files to effectively use the Army Information System.*

3. *A query formulation capability that allows users to express a formal request against one or more databases or files identified during browsing. This capability allows users who are unfamiliar with either the syntax and/or semantics of the language supported by a particular DBMS to effectively use the databases and files.*
4. *A distributed query processing capability that provides for the optimization and execution of multisite queries. This capability allows users to easily access and manipulate data residing in more than one database and/or file in a location-transparent way.*
5. *A multilevel security capability that provides for the administration and control of the information residing in ANSWER at multiple sensitivity levels. This capability allows users possessing a range of clearances to access data in ANSWER that spans multiple sensitivity levels.*

By providing these capabilities in a single uniform environment, ANSWER will:

1. *Facilitate the use of standards, procedures, and concepts described in the Army Information Model,*
2. *Enable Army personnel to make informed decisions through the utilization of large amounts of information,*
3. *Enable Army personnel with little training to formulate formal requests to a DBMS,*
4. *Assist Army personnel in the processing of formal requests on remote machines.*

1.3 Program Accomplishments

The original vision for ANSWER has been realized. The work has been accomplished in four phases numbered I to IV, with phase III split into subphases IIIA and IIIB. The ANSWER prototype has evolved over the phases as described below. Subsection 1.4 gives an overview of the ANSWER prototype as of the end of Phase IV.

1.3.1 Phase I (May 17, 1988–May 16, 1989)

The major accomplishments of Phase I include:

- Requirements definition.
- System design including information architecture, software architecture, and typical hardware configurations.

- Selection of a representation model—the Entity Category Relationship (ECR) model—for representation of database schemas. The ECR model has been used in database research at Honeywell since 1981; other similar data models (e.g., IDEF1X and enhanced versions of the Entity Relationship model) are now more commonly used [Bati92].
- Selection of the implementation environment: Sun-3 workstation, SunOS version 3.2, the C programming language (later LISP was also used), a prototype implementation of IRDS over Oracle from the National Bureau of Standards¹ and X-Windows.
- Implementation of an “encyclopedia manager,” including the IRDS implementation for the storage of individual schemas, integrated schemas and the Army Data Dictionary (ADD); a schema integrator; and input and output schema processors that provide communication between IRDS and the schema integrator.
- A browser for viewing and navigating the ADD and individual ECR schemas in graphical form and for creating, modifying and viewing associations between ADD elements (information classes, subject areas, prime terms and standard data elements) and individual schema objects (entities, categories, relationships and attributes).

1.3.2 Phase II (June 16, 1989–December 15, 1989)

The major accomplishments of Phase II include:

- Implementation of a uniform X-Windows-based graphical user interface for the ANSWER tool set. The user interface provides consistent information presentation and tool invocation facilities for end users, and facilitates addition and removal of tools from the ANSWER tool set.
- Modification of the schema integrator and browser to use a uniform graphical user interface. The schema integrator previously used a full-screen character-based interface; the browser used a graphical interface.
- Addition of new browser functions to support the creation of new schemas and modification of existing schemas.
- Development of a prototype data element creation tool (DECT) to assist the user in generating data element names in accordance with Army regulation 25-9. DECT ensures that the new names have the syntax defined by Army regulation 25-9, assists the user in selecting data element prime words and class words that have the correct meaning, and identifies existing data element names that might be used instead of the new data element name.

¹ Now known as the National Institute for Standards and Technology.

1.3.3 Phase IIIA (December 16, 1989-September 30, 1990)

The major accomplishments of Phase IIIA include:

- Development of a query formulator tool to assist the end user in creation of syntactically correct SQL queries for execution against individual databases. The user may formulate a query directly in terms of the relational model (tables, columns, etc.) as defined by SQL or in terms of ADD standard data elements. In the latter case, the query is translated to SQL prior to execution. The query formulator has special features for piecewise construction of complex queries.
- Investigation of several query processing algorithms and implementation of two of them. The first algorithm, a term substitution algorithm, is appropriate for queries against elements of an integrated schema. The second algorithm, an "approximate" query processing algorithm, is an approach for processing queries that refer to prime terms from the ADD for which there are no exactly corresponding entities in the database schema.

1.3.4 Phase IIIB (October 1, 1990-March 30, 1991)

The major accomplishments of Phase IIIB include:

- Modification of the IRDS implementation so that it may be used with a variety of UNIX file directory structures rather than a predefined directory structure.
- Port of the ANSWER software to SunOS 4.03, Oracle 6.0 and a new version of Common LISP/Common Windows.
- Identification of security issues peculiar to heterogeneous distributed database environments such as ANSWER.
- Identification of problems that arise when processing queries against an integrated schema when the underlying individual schemas have incompatible attribute domains, entities, or levels of detail.
- Implementation of a distributed query processing capability that supports access to databases stored in remote Oracle and Informix databases.
- Demonstration of the ANSWER concept for database integration as an alternative to the approach taken in the ILAPS project. ILAPS (Installation Level Acquisition of Products and Services) integrates supply, contracting and finance data systems by defining transfers of data between the systems; the end user may access only one system at a time. In contrast, the ANSWER approach is to define an integrated schema incorporating the individual data systems. Fragments of the supply data system (called SAILS) and contracting data system (called SAACONS) were added to the ANSWER system, stored under Oracle and Informix, respectively.

1.3.5 Phase IV (July 1, 1991-March 31, 1992)

The major accomplishments of Phase IV include:

- A major revision of the user interface design and implementation to make the ANSWER tool set easier to use. The user interface paradigm is now very similar to that of Microsoft Windows or Apple Macintosh.
- Addition of a function to convert a relational schema definition (expressed as a collection of SQL CREATE TABLE statements) to an Entity Relationship schema (i.e., an ECR schema with no categories defined.) The conversion is entirely automatic and implicit when opening a file that contains the relational schema definition. Features could be added to allow the user to identify certain entity sets as categories of other entity sets.
- Revision of the query formulator to make it easier to use. The original query formulator had features to assist the user in piecewise formulation of complex SQL queries. However, these features are somewhat distracting when composing simple queries, which are the most common. The new query formulator supports composition of SELECT-FROM-WHERE queries; table and column names can either be typed in or selected from a list of those defined in the schema for the database being queried.

1.4 Current ANSWER Prototype Functions and Architecture

Figure 1-1 illustrates forward and reverse database engineering and the process of accessing an integrated collection of databases. The shaded boxes indicate functions that the current ANSWER prototype provides to support these activities.

Forward engineering is the process of designing and implementing a database. The database administrator designs the schema using a conceptual model such as the ECR model. The conceptual schema is then converted to an implementation schema expressed in the data model of the database management system, often the relational model. The ANSWER system provides a graphics-based tool for designing ECR conceptual schemas. It does not provide a function for converting a conceptual schema to an implementation schema, as this is currently somewhat DBMS-specific and creation of new databases is not the primary focus of the ANSWER project.

Reverse engineering is the process of retrieving a conceptual schema from an existing implementation schema. In the context of ANSWER, the database administrator uses reverse engineering to produce conceptual schemas from multiple existing databases as input to the schema integration process. Again, ANSWER does not actually extract a relational schema from a DBMS, as this is currently DBMS-specific. Instead, ANSWER accepts a relational schema definition in the form of a text file that uses standard SQL table definition syntax.

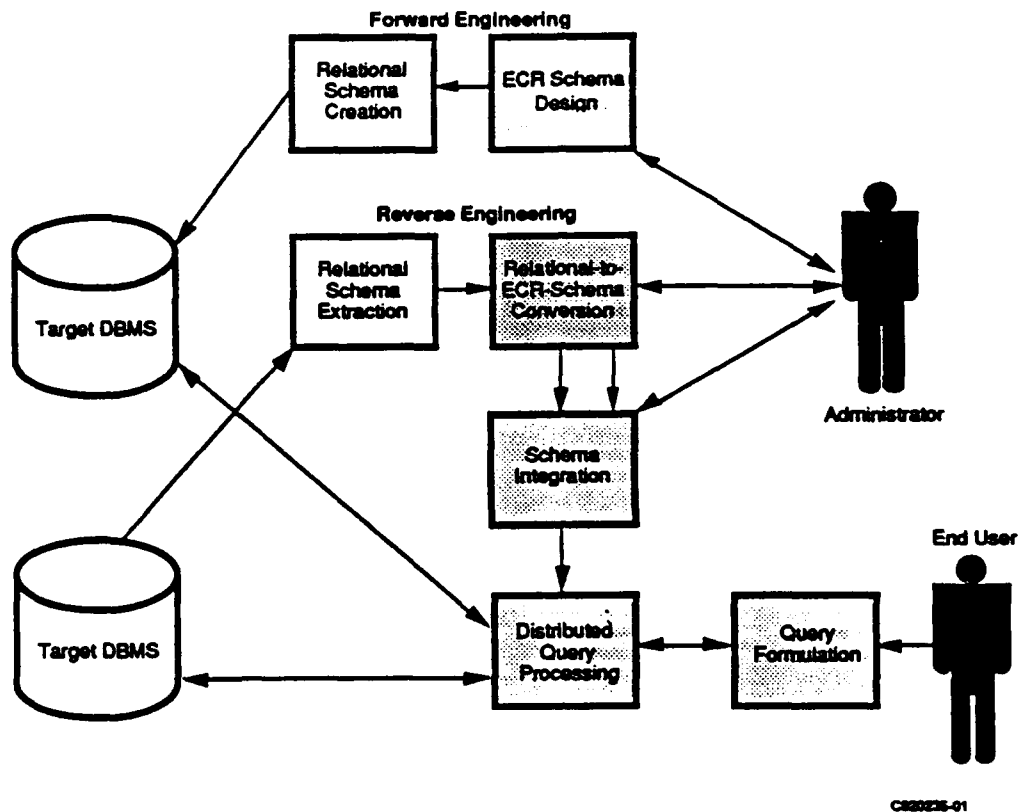


Figure 1-1. Engineering and Accessing Databases

The database administrator uses the ANSWER schema integrator to generate a single integrated schema from two individual database conceptual schemas. This is a highly interactive process that requires the database administrator to have a deep understanding of the meaning of the data in the databases that must be integrated. Schema integration results in an integrated schema against which the user can formulate queries and a set of mappings between the integrated schema and the individual database schemas.

The end user accesses an integrated database by formulating a query expressed in SQL. The ANSWER prototype provides an interactive query formulation tool to assist this process. It also provides a distributed query processor to decompose the query into subqueries on the constituent underlying databases and to combine the results of these subqueries for presentation to the end user. The decomposition process is guided by the mappings produced by the schema integrator.

Figure 1-2 shows the architecture of the ANSWER system. The database administrator and end user have access to ANSWER functions via a common graphic user interface. The user can create, modify and view ECR conceptual schemas using the browser function. Conceptual schemas can be stored in either IRDS or UNIX files. The browser includes the relational-to-ECR conversion function that automatically converts relational schema descriptions when the user selects them for browsing. The schema integrator, query formulator, and distributed query processing functions are as described above. ANSWER includes Oracle and Informix database

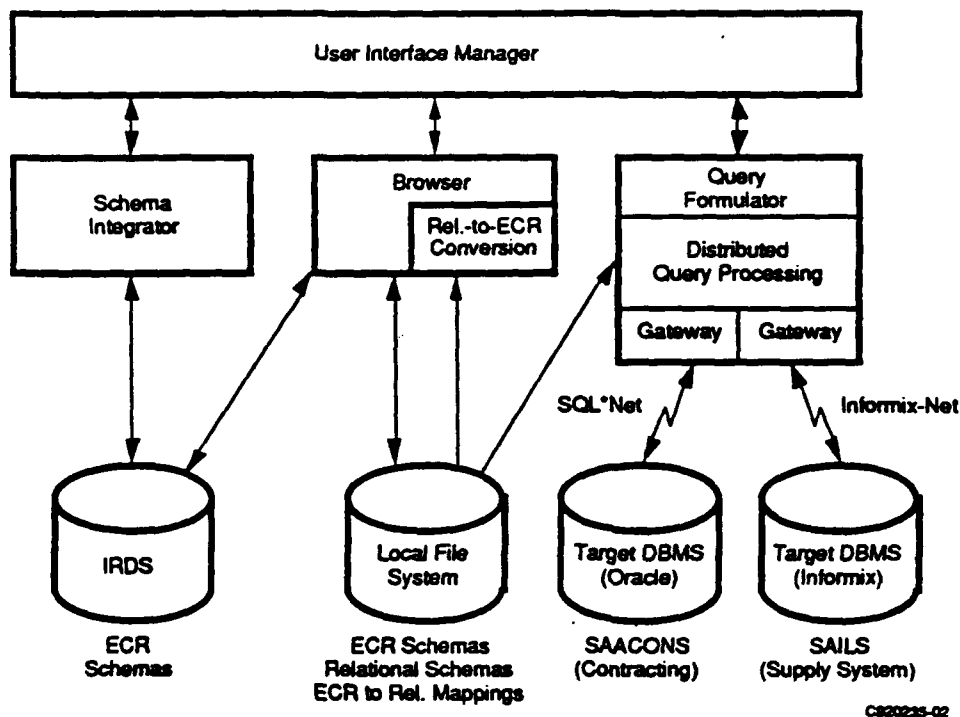


Figure 1-2. Architecture of Current ANSWER Prototype

management systems (DBMSs) to store sample databases, including extracts from two Army databases, SAACONS (a contracting database) and SAILS (a supply system database). The application program interfaces and network protocols for accessing Oracle and Informix databases are currently different, so the distributed query processor includes simple gateways that hide those differences from the rest of the system.

1.5 Standards-Based ANSWER Architecture

Current developments in the database standards and vendor communities will make a standards-based, vendor-independent ANSWER architecture implementable in the near future. Those developments include:

- Continued evolution of the ISO/ANSI SQL standard,
- Development of the ISO/ANSI OSI Remote Database Access (RDA) standard,
- Development of the SQL Access Group SQL and RDA specifications.

To date, DBMS vendors have implemented different dialects of SQL in their products. Several vendors now comply with the current SQL standard (ISO 9075:1989), but the functions included in that standard are so limited that many applications must rely on vendor-specific features. A new version of the SQL standard, informally called SQL2, is expected to be approved this year. SQL2 incorporates significantly more functionality than the current SQL standard, so that many applications will require no vendor-specific SQL features.

The first version of the RDA standard is also expected to be approved this year. RDA, which belongs to the OSI suite of standards, defines a client-server application layer protocol for accessing SQL databases across a network. To date, such protocols have been vendor-specific, such as Oracle's SQL*Net and Informix's Informix-Net.

When products become available that conform to these standards, implementation of the standards-based ANSWER architecture shown in Figure 1-3 will be feasible.

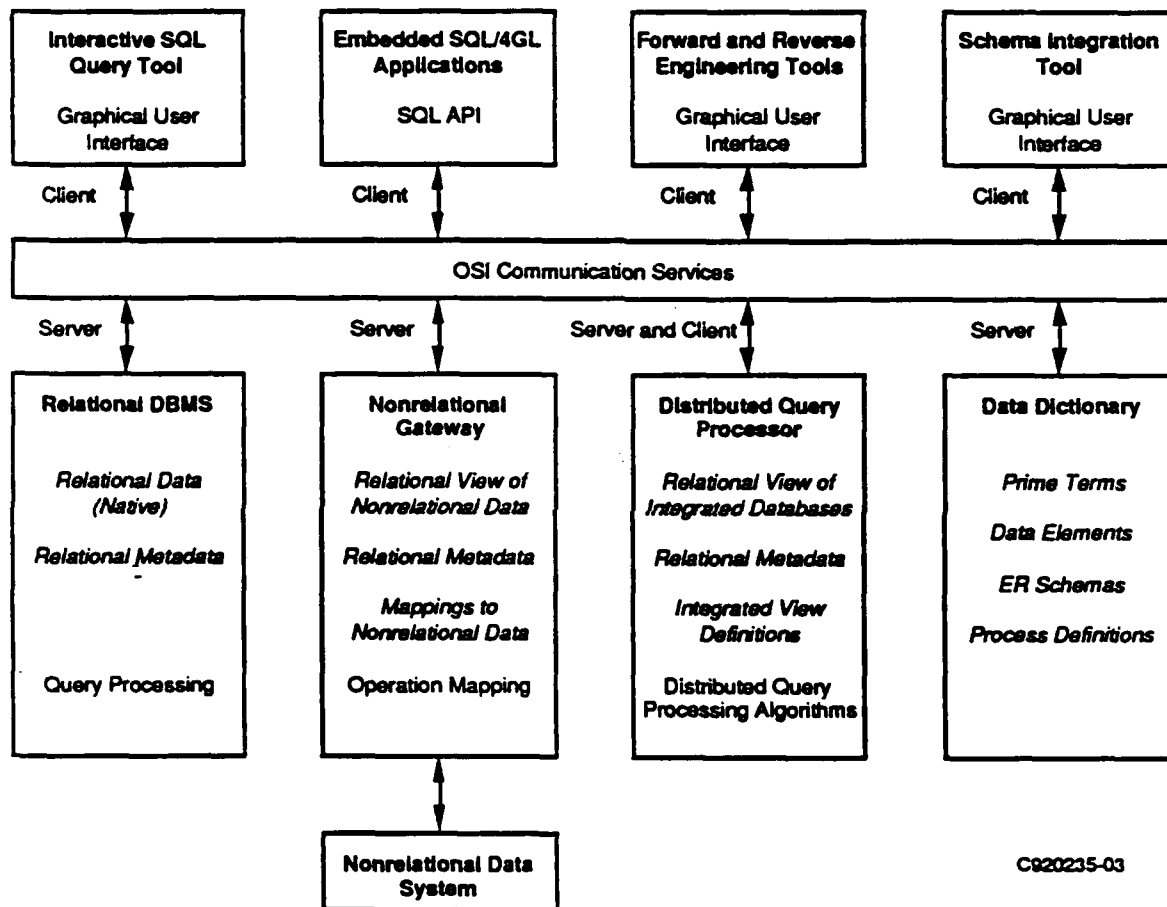


Figure 1-3. Standards-Based ANSWER Architecture

It is a client-server architecture. Clients include interactive SQL query tools, applications developed using embedded SQL or 4GL tools, forward and reverse engineering tools, and a schema integration tool. Servers include relational DBMSs, gateways that permit access to flat files and other nonrelational data systems using SQL, a distributed query processor, and a data dictionary. Servers all accept standard SQL as a data definition and manipulation language. Clients submit SQL statements to servers and receive results via RDA and lower-layer OSI communication services. The distributed query processor acts both as a server that accepts queries from end users and a client that submits subqueries to other servers.

All of these elements except the schema integration tool are currently available in proprietary form. Vendors will make RDA-compliant versions available when the standards are in place and sufficient customer demand is present. The next version of the U.S. Government Open Systems Interconnection Profile (GOSIP) will include RDA; this should generate significant demand for RDA-compliant products from U.S. government agencies.

Some forty vendors of database management systems and related software have formed the SQL Access Group (SAG). This group has worked in parallel with the SQL and RDA standardization process. It has developed its own interim specifications of RDA and a vendor-neutral dialect of SQL that includes many SQL2 features. These specifications have been published through X/Open. SAG has stated its intention to migrate its specifications to the ISO/ANSI RDA and SQL specifications when those specifications become standards.

SAG has also demonstrated multi-vendor interoperability among clients and servers adhering to these specifications and has defined a version of RDA that works over TCP/IP networks. Several vendors have announced products adhering to the SAG specifications and more announcements are expected in the near future. An interim implementation of the standards-based ANSWER architecture could incorporate products adhering to these specifications. Vendors of these products can be expected to provide means for migrating to ISO RDA/SQL-compliant products.

1.6 Recommendations for Future Activities

There are two activities that the Army could undertake to facilitate the implementation of ANSWER functionality at Army operational sites:

- Implement the standards-based ANSWER architecture outlined above for evaluation and operational use at an Army end-user site. The primary objective would be to evaluate the degree to which such an assemblage of commercially available products meets the heterogeneous database access needs of Army end users and to identify further research and development tasks. A secondary objective would be to encourage product vendors to accelerate introduction of standards-compliant products.
- Conduct further research in schema integration technology, emphasizing prototyping and evaluating schema integration techniques in the standards-based architecture. As mentioned above, the authors are not aware of any commercial schema integration products that meet the needs of the ANSWER architecture. The main roadblock is the definition of sufficiently flexible techniques for resolving schema-level and instance-level differences in the way two databases model the same aspects of the real world. Such work should leverage more basic research in schema integration and related issues sponsored by the Army Research Office, Rome Laboratory and other government agencies.

Section 2

Functional Requirements of the ANSWER System

The ANSWER system is a set of tools to aid data administrators and end users in managing and accessing data from distributed and heterogeneous databases, to improve the usefulness and effectiveness of diverse information systems, and to improve user access to these systems by enhancing interoperability, integration and synchronization. These improvements are to be achieved without requiring users to understand much of the local logical structures and the physical locations of the individual databases, system-specific procedures for accessing the data, and procedures for combining data from various sources.

Basically, ANSWER provides the following functionalities: integration of schemas from different databases, viewing of data with individual or integrated schemas, and formulation and execution of user queries based on integrated schemas.

This section describes the functional requirements of the ANSWER system. It includes descriptions of functional components, hardware and software configurations, and data models used.

2.1 Functional Components

ANSWER is divided functionally into the following components, based on the basic functionality requirements as alluded to above (see Figure 1-2):

- **The User Interface Manager (UIM)**—The UIM is a multi-window graphical user interface that supports control integration of multiple tools. The UIM controls the tool interfaces, managing input and output for each tool as well as window managing functions and graphical display functions required by the tools.
- **Schema Integrator**—The schema integrator supports the interactive creation of integrated views of data contained in the databases under ANSWER and automates the production of integrated schemas from individual schemas.
- **The Information Resource Dictionary System (IRDS)**—IRDS is used here to store data dictionaries and individual or integrated schemas represented in ANSWER.
- **Browser**—The browser provides users access to the databases under ANSWER through commands, menu choices, and data item selection and displays, modifies, and manipulates data and schemas according to the user commands.
- **Query Formulator**—The ANSWER query formulator is a tool designed to aid end users in creating well-formed SQL queries. It produces queries executable by an SQL processor.

- Query Processor—The query processor handles the processing of queries against heterogeneous databases.
- Relation-to-ECR Converter—The converter converts a relational schema to an ECR schema using semantic information specified by users.

The entire system runs on Sun 3 series workstations with SunOS 4.0.3. Three Suns can be networked together: one principal machine for the graphic user interface and ORACLE DBMS, a second machine for ORACLE databases, and a third one for INFORMIX databases. ANSWER can also run on a single Sun workstation.

Commercial and public-domain software packages needed include:

- Allegro Common LISP and Common Window on X,
- ORACLE DBMS,
- INFORMIX DBMS,
- X11 R4 and twm window manager,
- Gnu emacs.

2.2 Detailed Description of Data Models

This section defines the data models and data definition languages used in ANSWER.

2.2.1 ECR Model

This subsection presents the definition of ECR schemas and their representations in local files and IRDS.

ECR Schemas—The ECR schemas are described by a general ECR model with some ANSWER-specific constraints. The ECR model considered here is based the original definition presented in [Elma80] and [Elma85].

ANSWER imposes the following constraints on the ECR definition:

- Category is defined as a subclass of an entity.
- Only *binary* relationships are considered.
- A relationship may not have attributes.
- There are one-to-many (1:N) and one-to-one (1:1) relationships, but no many-to-many relationships. Assume that there are two entities *Ea* and *Eb* and that the relationship between *Ea* and *Eb* is *R*. Table 2-1 lists permitted types of relationships and their cardinalities.

Table 2-1. Correspondence Between Relationship Type and Cardinalities

Type	Dependency	Cardinality of Ea		Cardinality of Eb	
		min	max	min	max
1:N	Partial, Ea on Eb	>0	∞	0	1
1:N	Partial, Eb on Ea	0	∞	1	1
1:N	Total	>0	∞	1	1
1:1	Partial, Ea on Eb	1	1	0	1
1:1	Partial, Eb on Ea	0	1	1	1
1:1	Total	1	1	1	1

Representation of ECR Schemas in IRDS—Information Resource Dictionary System (IRDS) is a standard software tool for the management of data and information resources [NBSIR88-7300]. In ANSWER, IRDS is used to record and manage ECR schemas as well as other information for ANSWER tools. ECR schemas are represented in IRDS as follows.

To store ECR schemas in IRDS, the information about the ECR schemas (i.e., the meta-data) must be first defined by a Information Resource Dictionary (IRD) schema. The IRD schema (Figure 2-1) is based on the Entity-Relationship (ER) model [Chen76]. It includes entity types, relationship types, and attribute types. These types are specified as meta-entities in the schema. An IRD schema for the ECR meta-data information is defined using an IRD schema manipulation statement “add” (e.g., add meta-entity and add meta-relationship).

Once the IRD schema for ECR schema information (the meta-data) is defined, individual ECR schemas can be stored in IRDS by adding entities and relationships to the IRD schema.

Representation of ECR Schemas in Local Files—In ANSWER, the local file that stores ECR schemas is called an OF-AF-RF-DF list file. The file contains files of objects, attributes, relationships, and categories. The schema representation in the file has the syntax given in Figure 2-2.

2.2.2 Relational Model

This subsection describes the relational model used in ANSWER.

Relational Schemas—The relational schemas are described by a general SQL model with some ANSWER-specific constraints. The relational model is defined by a subset of ISO SQL2 Data Definition Language [ISO/IEC 9075]. Figure 2-3 gives the syntax for an SQL table definition.

The relational model is further constrained by ANSWER as follows:

- FOREIGN KEY is a singleton (i.e., it consists of only one table column name).
- In FOREIGN KEY specifications, the referencing column name and the referenced column name are always identical.

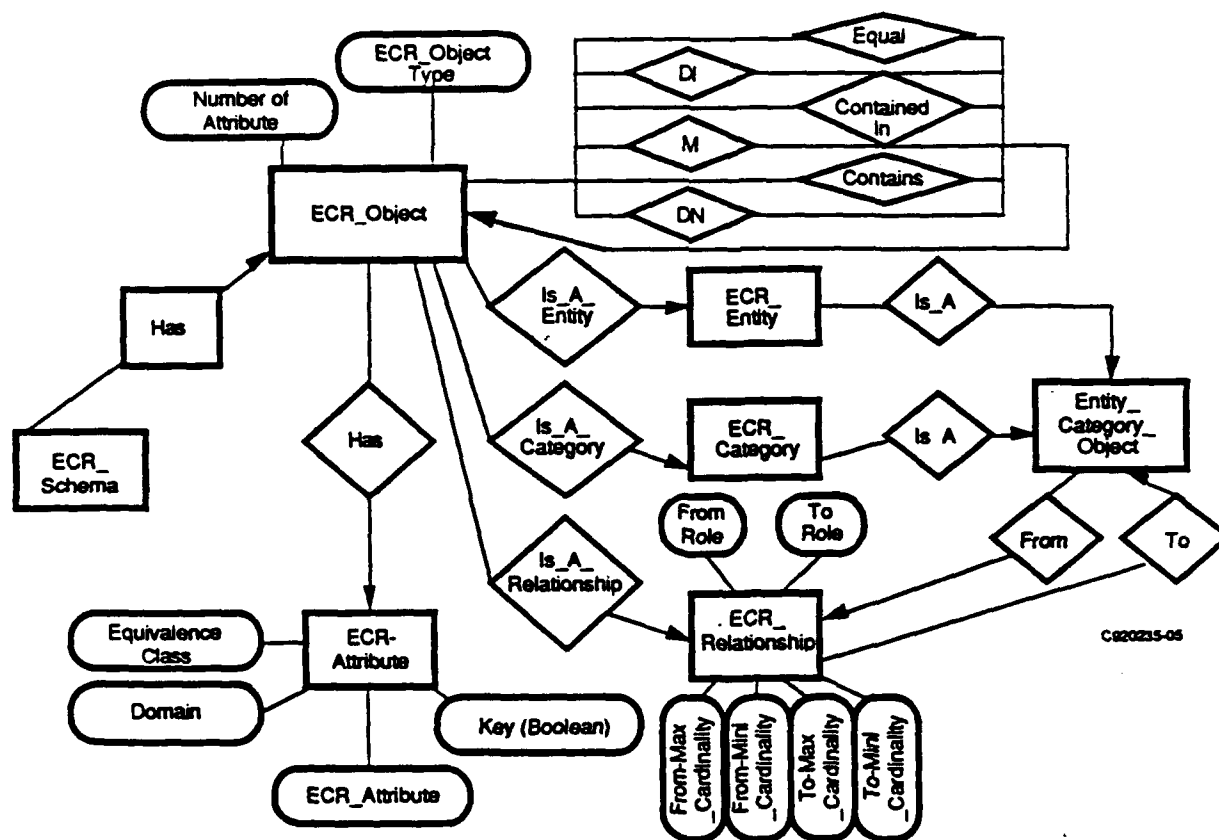


Figure 2-1. IRD Schema for ECR Schema Information

Representation of Relational Schemas in Local Files—A relational schema may be stored in a local ASCII file. The schema representation in local files shall follow the syntax specified in <table definition>, starting with CREATE TABLE and ending with EOF.

Data Manipulation Language—Figure 2-4 gives the syntax a subset of ISO SQL2 Data Manipulation Language [ISO/IEC 9075] that is used by the ANSWER system.

The query expression specified above shall follow the following syntax rules:

- All <column specification>s must refer to columns of tables listed in the FROM clause. If the column name appears in more than one table, the column specification must include an explicit table name to disambiguate the reference.
- For simplicity, the syntax does not allow table correlation names, so a table name may not appear more than once in a FROM clause.
- A quote character within a <character string literal> is represented by a double quote mark.
- A <column name> must be unique within a <table definition>.

```

<ecr> ::= ( setf <schema name>
          '((BROWSER_SCHEMA_INFO)
          <oflist> <aflist> <rflist> <dflist>
          EOT )

<oflist> ::= ( OFLIST [ { ( <object> ) }... ] )

<object> ::= <object_name> <object_type> { NO | <prime_term> }

<object_type> ::= E | C | R

<aflist> ::= ( AFLIST [ { ( <object_name> <attribute> ) }... ] )

<attribute> ::= <attribute_name> <attribute_type> <key> { NO | <standard_data_item> }

<key> ::= YES | NO

<rflist> ::= ( RFLIST [ { ( <relationship> ) }... ] )

<relationship> ::= <relationship_name> <objectA_name> <objectB_name>
                  <cardinalityAmin> <cardinalityAmax> <cardinalityBmin> <cardinalityBmax>
                  <roleA> <roleB> <objectA_type> <objectB_type>

<cardinalityAmin> ::= 0 | 1 | 2

<cardinalityAmax> ::= 1 | 3

<cardinalityBmin> ::= 0 | 1

<cardinalityBmax> ::= 1

<dflist> ::= ( DFLIST ( < category> [ <category> ... ] ) )

<category> ::= <objectA_name> <objectB_name>

```

Notes: 2 represents >0 in Table 2-1
3 represents ∞ in Table 2-1
object B is a subclass of object A

Figure 2-2. OF-AF-RF-DF File Syntax

2.2.3 ECR to Relational Schema Mapping

An ECR schema may correspond to an individual relational schema if it was converted from that relational schema directly, or to a collection of relational schemas if it was integrated from the set of relational schemas. In either case, a mapping exists from an ECR schema to its corresponding relational schema(s). This subsection describes such a mapping.


```

<table definition> ::=
    CREATE TABLE <table name> (
        <column definition> [{, <column definition>}...]
        [{, <constraint definition>}...]
    )

<column definition> ::=
    <column name> <data type> [NOT NULL]

<data type> ::=
    CHARACTER(<length>)
    | CHAR(<length>)
    | NUMERIC(<precision> [, <scale>])
    | DECIMAL(<precision> [, <scale>])
    | DEC(<precision> [, <scale>])
    | INTEGER
    | INT
    | FLOAT(<precision>)
    | REAL
    | DOUBLE PRECISION
    | DATE
    | TIME(<time precision>)

<constraint definition> ::=
    UNIQUE ( <column list> )
    | FOREIGN KEY ( <column list> )
      REFERENCES <table name> ( <column list> )

<column list> ::=
    <column name> [{, <column name>}...]

<table name> ::= <identifier>

<column> ::= <identifier>

<identifier> ::=
    <upper case letter> [[[<underscore>] <letter or digit>]...]

<letter or digit> ::= <upper case letter> | <digit>

<length> ::= <unsigned integer>

<precision> ::= <unsigned integer>

<scale> ::= <unsigned integer>

<unsigned integer> ::= <digit>...

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<upper case letter> ::=
    A|B|C|D|E|F|G|H|I|J|K|L|M
    N|O|P|Q|R|S|T|U|V|W|X|Y|Z

```

Figure 2-3. Syntax for SQL Table Definition

```

<query specification> ::=
    SELECT <column specification> {, <column specification>} ...
    FROM <table name> {, <table name>} ...
    [WHERE <search condition>]
    [ORDER BY <column specification> {, <column specification>}]

<column specification> ::=
    [<table name>.] <column name>

<search condition> ::=
    <boolean term>
    | <search condition> OR <boolean term>

<boolean term> ::=
    <boolean factor>
    | <boolean term> AND <boolean factor>

<boolean factor> ::=
    [NOT] <boolean primary>

<boolean primary> ::=
    <predicate>
    | ( <search condition> )

<predicate> ::=
    <value expression> <comp op> <value expression>

<value expression> ::=
    <column specification>
    | <character string literal>
    | <numeric literal>

<comp op> ::=
    = | < | > | <= | >=

<character string literal> ::=
    '<character>...'

<numeric literal> ::=
    <exact numeric literal>
    | <approximate numeric literal>

<exact numeric literal> ::=
    [+|-]{<unsigned integer>[.<unsigned integer>]
    | <unsigned integer>
    | .<unsigned integer>}

<approximate numeric literal> ::=
    <exact numeric literal>E[+|-]<unsigned integer>

```

Figure 2-4. ANSWER Subset of SQL Data Manipulation Language

Mappings to Individual Relational Schemas—In ANSWER, there is an implicit one-to-one mapping between a given ECR schema and its corresponding relational schema, as follows:

- An *entity* in an ECR schema is directly mapped into a relational table.

- A *category* in an ECR schema is also mapped into a relational table.
- *Attributes* of an entity or category are mapped to columns of the corresponding relational table.
- A *relationship* (one-to-many, one-to-one, or ISA) between entities or between entity and category is mapped to an identical column name (foreign key) of two corresponding relational tables.

Mappings to a Collection of Relational Schemas—The mapping from an integrated ECR schema to its corresponding collection of relational schemas is direct. In other words, there are no such multilevel mappings as from an integrated ECR schema to an integrated relational schema and then from the integrated relational schema to a set of existing individual schemas; integrated relational schemas do not exist in ANSWER.

Mappings to a collection of relational schemas are similar to those to individual schemas, where entity and category are mapped into relational tables and relationships are mapped into foreign keys. In addition, special care must be taken for the integrated part of the ECR schema; specifically:

- Each entity that was derived as a generalization of different categories is mapped respectively to individual relational tables that correspond to the categories. Note that the relational tables are defined separately in two or more relational schemas that participate in the schema integration.
- Each relationship that involves a derived object (entity or category) specifies its relationships, through *global foreign keys*, with other objects (entities or categories) that also involve the same derived object.

Representation of Mappings in Local Files—The mapping information between an (integrated) ECR schema and its corresponding relational schema(s) is stored in a set of local files. The file formats and mapping rules are described in the following paragraphs.

Mappings for an Integrated ECR Schema—Five types of files are used for the mapping. They are REL_MAP_{*i*}, ATTR_MAP_{*i*}, FKEY_{*i*}, GLOBAL_KEY, and REMOTE_INFO, where *i* is the number of relational schemas participating in the schema integration:

- REL_MAP_{*i*}—contains the information of mapping between the ECR entities/categories and the tables of relational schema *i*. The file format and mapping rule are:

<ECR entity/category name> = <table name>

- ATTR_MAP_{*i*}—contains the information of mapping between the entity/category attributes and the table columns of relational schema *i*. The file format and mapping rule are:

<ECR attribute name> = <table name> <column name>

- **FKEY_i**—contains the information of mapping between the ECR relationships and the foreign keys of relational schema *i*. The file format and mapping rule are:

<referenced table name> <referencing table name> <foreign key>

- **GLOBAL_FKEY**—contains the information of mapping between ECR objects (entities or categories) that involve the same derived object. In other words, this file is used to show global join conditions between relations in different relational schemas. The file format and mapping rule are:

<table x of schema *i*> <table y of schema *j*>
 <a column name of table x> <a column name of table y>

- **REMOTE**—contains the information about relational schemas and the databases on which they reside. The file format is as follows:

<relational schema name> <database type> <server information>

<database type> ::= oracle | informix

<server information> ::= <pathname> *for oracle* |
 <accountname password hostname sid> *for informix*

Mappings for a Single ECR Schema—Mappings for a single ECR schema share a subset of local files defined for mapping of integrated ECR schemas. Since no schema integration is involved here, GLOBAL_FKEY mapping file is not necessary. For the same reason, only single REL_MAP, ATTR_MAP, and FKEY files are used.

2.3 Detailed Description of Functional Components

This section presents detailed descriptions of all functional components listed before.

2.3.1 IRDS

The Information Resource Dictionary System (IRDS) was initially developed by the National Institute for Science and Technology [NBSIR 88-3700] and is used here to store data dictionaries and individual or integrated schemas represented in ANSWER.

IRDS runs on top of ORACLE DBMS. Information in IRDS is stored in relational schemas in ORACLE. All IRDS commands are translated into SQL commands for ORACLE.

IRDS uses ER data models. Schemas represented in ANSWER are in ECR models, and therefore should be translated into the ER model [Elma85]. Figure 2-3 depicts the ECR model as represented in IRDS in terms of the ER model.

The Browser stores schema definitions to IRDS and retrieves them when needed. The definitions from IRDS can be made available through the Browser to the Schema Integrator, Query Formulator, etc.

2.3.2 Browser

The Browser provides users access to the schemas of the databases under ANSWER. It displays and manipulates interactively graphical diagrams of ECR schemas, through commands, menu choices, and data item selection from users.

The browser provides the following functionalities:

- Open/save schemas,
- Display schemas,
- Edit schemas,
- Add/delete nodes,
- Index nodes by names,
- Edit node definitions, names and attributes,
- Find/display nodes,
- Scroll nodes,
- Create new links between nodes,
- Define relationships between nodes,

Here, "node" refers to entities, categories, or relationships in data schemas. These functionalities are available through global commands from the user interface, with interactive windows and graphics.

2.3.3 Schema Integrator

The schema integrator supports the interactive creation of integrated views of data contained in the databases under ANSWER and automates the production of integrated schemas from individual schemas. The schema integrator reads schemas from IRDS and interacts with users to obtain information about equivalence relationships between individual entities and attributes. Based on the information, it synthesizes a complete integrated description of the input schemas, which may include additional entities not present in the original schema as a result of the generalizations discovered during the integration process.

The integration process is illustrated by the schema integration script shown in Figure 2-5. The basic architecture of the schema integrator is shown in Figure 2-6.

The initial schemas reside in IRDS and can be retrieved by the schema integrator when needed. The integrated schema will reside in IRDS or local files according to the user selection.

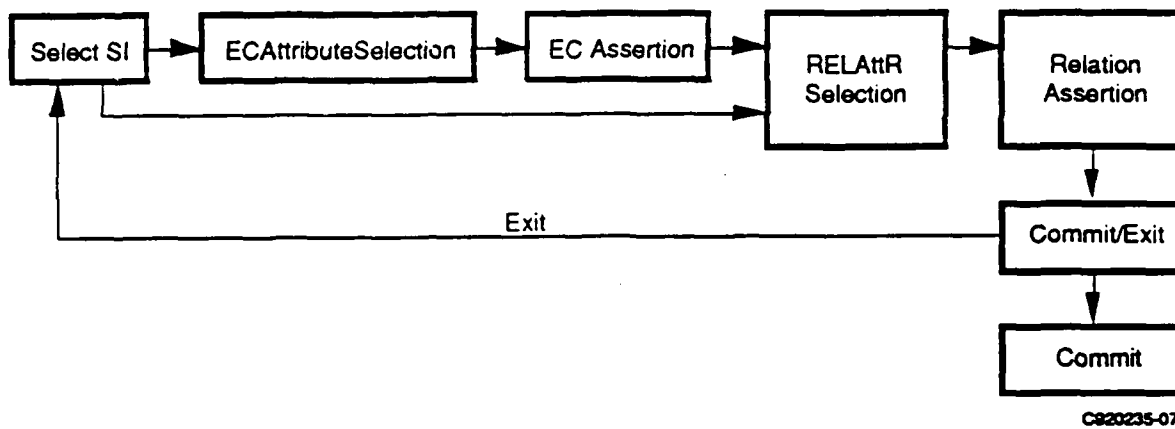


Figure 2-5. Schema Integration Script

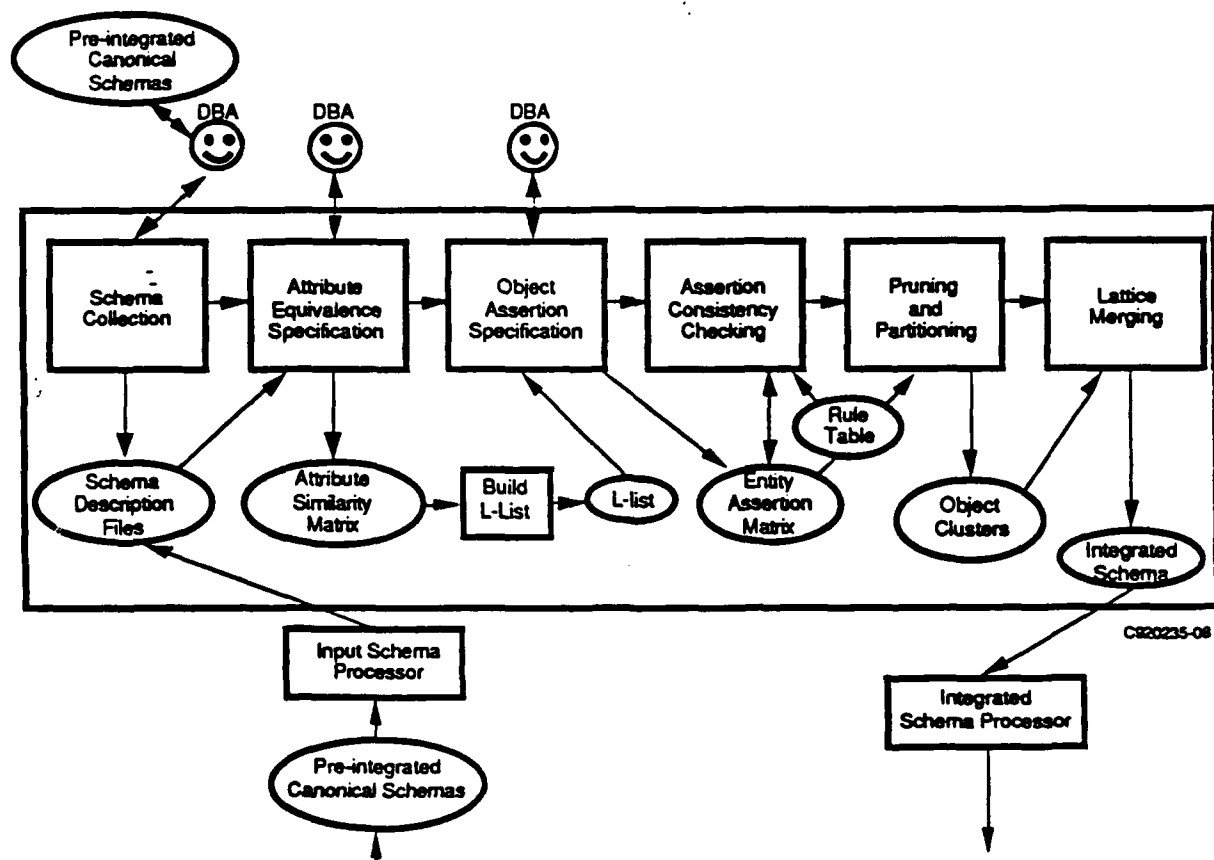


Figure 2-6. Schema Integrator Architecture

2.3.4 Query Formulator

The ANSWER query formulator is a tool designed to aid end users in creating well-formed SQL queries. It produces queries executable by an SQL processor.

Both graphical and textual means are provided to create and edit queries.

The query formulator should support a "modeless" environment, in which users can freely move between text and graphic modes of entering query terms. Users of the query formulator may enter or edit queries by selecting options from menus or typing information into the editing window.

2.3.5 Query Processor

The query processor handles the processing of queries against heterogeneous databases. The query processor will first determine if a query sent to it is a local query or a global query. In case of a global query, distributed query processing is involved. The basic steps of distributed query processing include:

- Decomposition of a global query into local queries,
- Execution of local queries at various sites,
- Retrieval of results from local queries,
- Combination of local queries results.

The query processor accepts queries from the query formulator, interacts with remote databases, and sends results back to the query formulator.

2.3.6 Relational-to-ECR Converter

The relational-to-ECR converter allows users to obtain an ECR schema, and view its diagram, for a given relational schema. In detail, the conversion function shall take a relational schema and convert it to an ECR schema. During the conversion process, semantic information may need to be provided from users in order to identify subclasses and create generalization hierarchies for the ECR schema. In addition, the conversion process shall generate an ECR-to-relational mapping. This mapping information will be used during query processing, where queries issued against the ECR schema need to be mapped to the corresponding relational schema. Figure 2-7 depicts the conversion function. The inputs and outputs of the conversion function are specified as follows:

- *Relational schema* is defined by the relational model.
- *ECR schema* is defined by the ECR model.
- *Semantic information* is provided by users and shall be able to specify ISA (subclass) relationships between entities.
- *ECR-to-Relational mapping*.

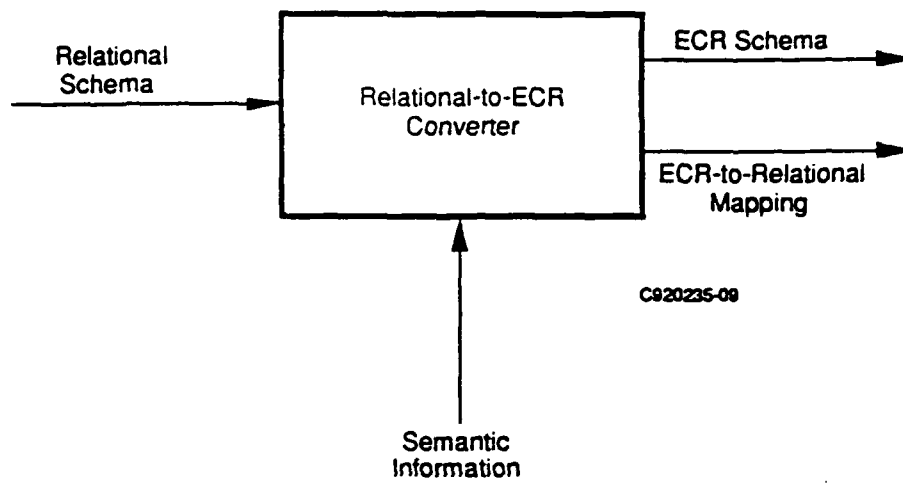


Figure 2-7. Relational-to-ECR Conversion Function

Section 3

User Interface to ANSWER

The user interface to the ANSWER software is presented in a windowed environment on a workstation high-resolution monochrome monitor. User interaction is based on mouse input, with direct manipulation of objects through point and click. It presents a user-driven environment: To control the flow of the interaction, the user selects pop-down menu options to request system actions, selects screen objects for interaction, and converses with the system through simple dialogs containing selectable textual lists and buttons. The system indicates to the user which choices are available at any given time.

The ANSWER interface comprises a main window that has a menu bar at the top and windows representing schemas, windows containing queries, dialog boxes for opening or editing schemas, and message boxes for system questions.

3.1 Interface Organizers

The visual space of the ANSWER interface is organized within a main ANSWER window through the use of four discrete types of sub-windows: Schema windows, Query windows, dialog boxes, and message boxes. Schema windows present visual representations of schema structures for browsing and editing schemas. Query windows present a structured environment to form queries on schemas. Dialog boxes present sets of parameters, settings, and objects for selection by the user to perform complex commands (e.g., editing a selected node). Message boxes provide for simple confirmation and system warnings and information, when necessary.

3.1.1 *Schema Windows*

Schema windows contain representations of schemas in the form of a connected graph of named rectangles representing nodes, with named relationships between nodes. Schema windows may be opened, closed, printed, or made active, and the schemas they represent may be saved under different names, deleted, or altered. Queries may be performed on the schema, and schemas may be integrated using the Schema window.

The nodes represented in the Schema window represent database entities and categories, and actions taken on the node representations are taken on the corresponding database objects. Nodes may be selected by the user for editing, new nodes may be added, and relationships between nodes may be added or altered. Nodes may be searched for by name.

3.1.2 *Query Windows*

Query windows are special sub-windows within the main ANSWER window that represent queries posed to the active schema. A Query window may be opened, closed, deleted, renamed, altered, or printed, and the query represented in the window may be executed. Query windows

have their own menu bar and contain three large text boxes for the entry of Select, From, and Where query clauses.

3.1.3 Dialog Boxes

User-system dialogs are presented, where required, within dialog boxes. These dialog boxes are bounded areas, centered on the screen, that contain simple interaction spaces to perform user inputs that cannot be specified with a simple command. For example, the dialog that supports opening a schema presents a list of the known schemas (so the user does not have to rely on memory), and allows the user to simply point to the name of the desired schema and click the mouse button, then point to a pushbutton labeled "Open" and click the mouse button. The dialog box closes, and the selected schema is opened. All dialog interactions are designed in this way, which provides direct manipulation of objects and commands, provides defaults and lists where necessary, and limits the available interactions to those required by the current context.

3.1.4 Message Boxes

Questions that the system must ask of the user are presented in message boxes. Message boxes are very simple dialog boxes, containing a question or statement from the system and a limited number of labeled responses for the user to select. Message boxes commonly request confirmation for user requests that are destructive, such as deleting a node or schema.

3.2 Dialog Flow in ANSWER

When the ANSWER software is invoked, only the Schema menu is available the user. After the desired schema is opened, the menu may be selected again to open additional schemas. Schema windows are made active through clicking on the window or by selecting the window name from the Windows menu. The Schema and Edit menus apply to the active schema. Nodes in the active window may be selected, making the Node menu active. The Node menu items apply to the active node. Only one schema and one node within that schema may be active at any one time.

Users may pose queries on the open schema by selecting "Query..." from the Edit menu. This causes a Query window to open on the screen, containing a structured query builder. Users may also select to integrate two schemas, by opening the first schema and making it active, selecting "Integrate..." from the Edit menu, then selecting the second schema from a standard open dialog. This is followed by a dialog that allows the user to assert equivalences for the two schemas.

3.3 Interface Components

The main interface components in the ANSWER user interface are the mouse cursor, which indicates the subject of the users interest, the menu bar, for selecting high level actions, various pushbuttons and text displays for selection of actions and objects, edit boxes for specifying text input from users, and static system text.

3.3.1 Mouse Cursor

The mouse cursor is always present on the screen. Its movement on the screen is controlled by the movement of the mouse input device. To select a selectable item displayed on the screen within the main ANSWER window, the user positions the mouse cursor over the object by moving the mouse on a horizontal surface, then presses the left button on the mouse. This is referred to as a mouse click, or simply as "clicking."

3.3.2 Menu Bar

The menu bar appears across the top of the main ANSWER window, starting from the left side of the window. The individual menu names appear at all times. To see the items associated with a menu, the user clicks the mouse button when the mouse cursor is over the menu name. A box containing the menu items appears below the menu name. As the user moves the mouse cursor over each item, the item is highlighted in inverse (white letters on a black background). If the user clicks over the menu item, the menu item list disappears, and the command named by the menu item is invoked. If the user clicks over an empty space within the window, the menu item list disappears and no action is taken.

Menu items are followed by an ellipsis (...) if they cause a dialog box or message box to appear. Menu items without an ellipsis represent simple commands. Groups of related items are separated within menus by an ellipsis on a line by itself. Any menu or menu item that is not applicable to the current context is disabled: it appears "grayed out," does not highlight when the mouse cursor moves over it, and is not selectable by the user (i.e., clicking over the item has no effect).

3.3.3 Pushbutton

Pushbuttons are used to command actions in dialog boxes and to enter choices in message boxes. Pushbuttons hilite (become visibly inverted, black for white and vice versa) when the user moves the mouse cursor over the pushbutton, to show that the pushbutton is selectable. Pushbuttons momentarily highlight when the mouse button is pressed when the mouse cursor is over the pushbutton to show that the action has been carried out. Pushbuttons are rectangular, and labeled with the name of the action that is invoked by the pushbutton. Dialog box pushbuttons that close the dialog ("Open," "Save," "Cancel") appear vertically stacked in the lower right corner of the dialog. "Cancel" always causes the dialog box to be closed with no action taken. Message box pushbuttons ("Yes," "No") appear along the bottom of the message box. Unavailable pushbuttons appear gray to the user, and may not be selected. Any pushbutton that is not relevant to the current interaction context is made unavailable by the system and may become enabled in the course of the interaction if the context changes.

3.3.4 Node Button

A form of pushbutton appears in the Schema window, representing a schema node. Node buttons are labeled with the name of the node. Node buttons representing category nodes are six-sided, elongated polygons, and node buttons representing entity nodes are rectangles. (Relation nodes do not have node buttons.)

If the user clicks the mouse button while the mouse cursor is over a node button, the node button becomes highlighted. If the node button is already highlighted, the node button becomes unhighlighted. Only one node button may be selected in a particular Schema window at one time. If a node button within the active Schema window is selected, the "Node" menu becomes enabled, and menu selections apply to the selected node.

3.3.5 Radio Button

Radio buttons are presented in mutually exclusive groups of two or more. They appear as open circles, followed by the name of the pushbutton. Only one radio button in a group may be selected at any time; a selected radio button has a filled circle inside the button circle. Groups of radio buttons are boxed with a thin black line.

3.3.6 Textual List

A textual list is a box containing a list of single-line text strings, with a scroll bar on the left of the list if the list is longer than the size of the box. Text strings are truncated to fit the list width; text wrapping does not occur. Textual lists have a black title bar above the list box, with the title reflecting the category of the list items (for example, a textual list of the names of the nodes in a schema is entitled "Nodes"). One item in the textual list may be selected by the user by pressing the mouse button when the mouse cursor is over the item's text line. Selection of an item results in the item becoming inverse highlighted. If there is an associated edit box or text box, the item also appears in the associated box. Only one item may be selected from the list at any time; the previously selected item becomes unhighlighted when another is selected. Unavailable items appear gray to the user, and may not be selected.

3.3.7 Edit Box

An edit box is a rectangular box, with a black title above it containing either the edit box title or instructions to the user (e.g., "Type Schema Name:"). Edit box titles and instructions are always followed by a colon. The user may type into the edit box.

3.3.8 Text Box

A text box is a rectangular box that generally appears in a dialog box below a textual list. When an item is selected in the textual list, that item appears in the text box. Users may not type into text boxes.

3.3.9 System Text

System text is unadorned text supplied by the system for the users information. Users may not interact with system text.

3.4 Interface Design

The design of the major components of the user interface are discussed below. These components include the main ANSWER window and its associated menus, the Schema windows, which represent the structure of schemas, dialogs to support the user-system interactions, and message boxes for confirmation.

3.4.1 Main Window

Table 3-1 lists the menus and menu items available in the menu bar of the main ANSWER window. The Schema menu (Figure 3-1) is always enabled; if no schemas are open, the only enabled menu items in the Schema menu are "Open Schema...", "New Schema...", "Delete Schema..." and "Quit." If at least one schema is open, all Schema menu items are enabled. The Edit menu is enabled when at least one schema is open, and one schema is active (the currently selected Schema window). The menu item "Query" is enabled if the Schema allows queries, and "Integrate" is enabled if integration is possible with that Schema. The menu "Node" and its menu items (Figure 3-2) are enabled if a node is selected in the active Schema window. The menu "Windows" is enabled if at least one Schema window is open, and contains the names of all open Schema windows. The currently active Schema window has a check mark before its name in the menu item list.

3.4.2 Schema Window

The Schema window (Figure 3-1) has a black title bar above the window, containing the name and type of the schema. If the schema has not been saved, the text "(UNSAVED)" is appended after the schema name. Schema windows contain networks of nodes and relations. Nodes are represented by node buttons, and relations are represented by system text. Connections between nodes in the network are represented by lines between various nodes and relations. Node buttons in the Schema window are selectable by the user. The Schema window may be moved and resized. A different view of the schema may be achieved by opening the schema map through the Edit menu item "Show Overview," and selecting the relevant portion of the map.

If one or more schemas are in use, they are presented within the main window inside of sub-windows. Each sub-window has a black title bar at the top, containing the title of the schema. The window body of a schema sub-window contains a visual depiction of the schema organization, in the form of a connected graph of labeled nodes and relationships connecting nodes. To open a schema, the user selects the Schema menu on the main window menu bar, and selects the "Open Schema..." or "New Schema..." menu item. The newly opened schema is automatically the active Schema window. If numerous Schema windows are open, the user clicks on a window to make it the active Schema window and bring it to the front (conceptually closest to the user on the screen). Only one Schema window may be the active schema; all inactive Schema windows have a gray title bar.

Table 3-1. Menus and Menu Items—Main ANSWER Window

Menu	Menu Items	Result of Selection
Schema	Open Schema...	Open Schema dialog
	New Schema...	Add Node dialog
	Close Schema	(active Schema window closes)
	Save Schema	(active schema saved)
	Save Schema As...	Save As dialog
	Print	(active Schema window printed)
	Delete Schema...	Delete? message
	Quit...	Quit? message
Edit	Add Node...	Add Node dialog
	Show Overview	(Overview window for active schema appears)
	Show Legend...	Legend message
	Resize	(resizing enabled; see Schema window description)
	Redraw	<active Schema window redrawn>
	Find Node...	Find Node dialog
	Query...	Open Query dialog
	Integrate...	Open Schema dialog
Node	Edit Attribute...	Edit Attribute dialog
	Edit Node...	Edit Node dialog
	Delete Me...	Delete? message
Windows	(name of open schema ₁)	(bring named Schema window to the front and make the active schema)
	(name of open schema _n)	

Within Schema windows, the user interacts with the schema components by selecting a node and using the Node menu on the menu bar to take actions on the node (e.g., editing the node or its relationships). Only one node may be selected in a Schema window at one time, and any actions or commands taken while the node is selected apply only to that node. Selection of a node causes the previously selected node to become unselected. A selected node appears inversely highlighted (white text on a black rectangle).

The user moves and resizes the active Schema window by selecting the Resize item in the Edit menu, positioning the mouse cursor to the desired position of the Schema window's upper left corner, clicking, repositioning the mouse cursor to the desired position of the Schema window's lower right corner, and clicking again.

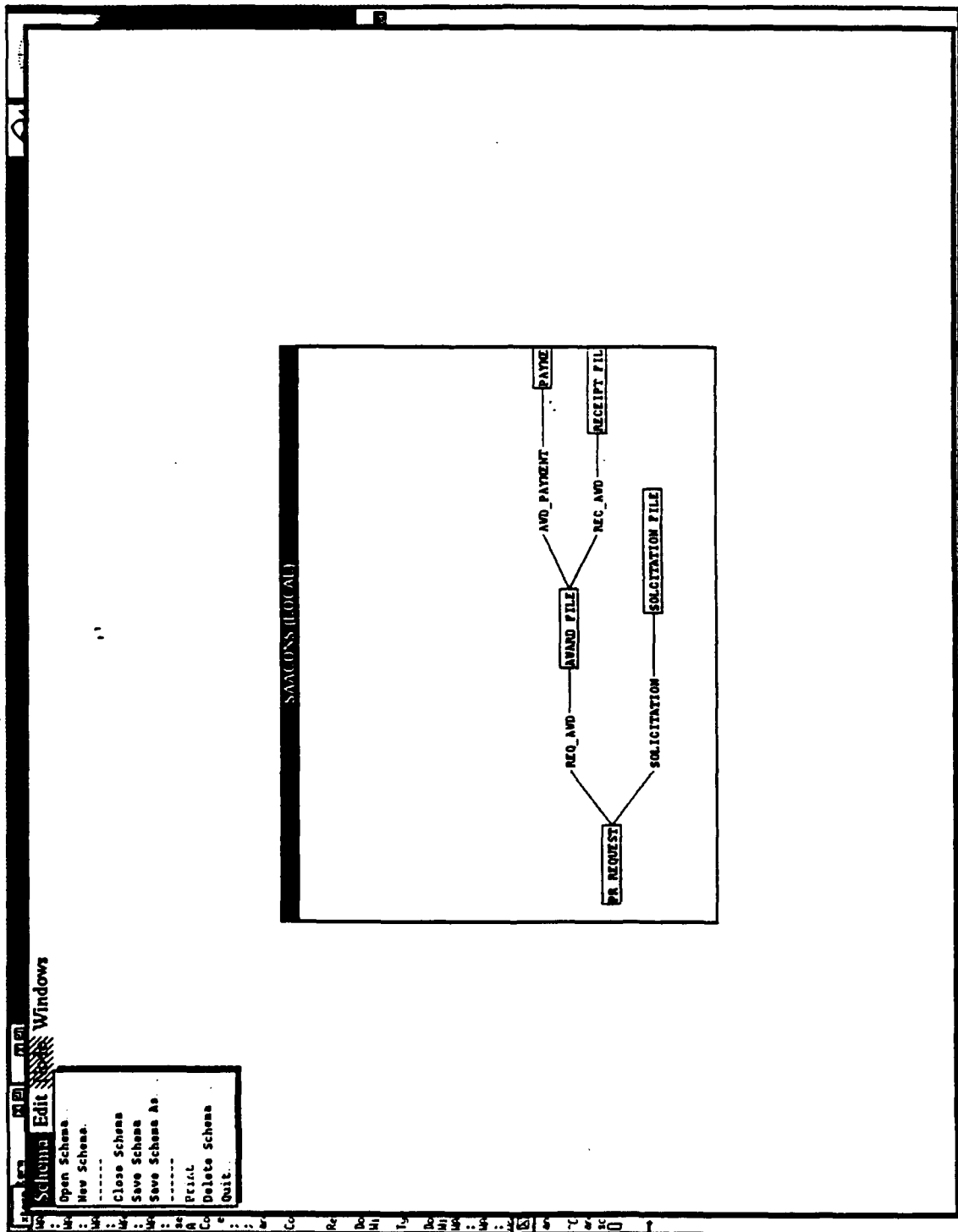


Figure 3-1. Main Window: Schema Menu

3.4.3 Query Window

Table 3-2 lists the menus and menu items available on the menu bar of the Query window. Query windows are opened through the Schema menu of their associated schema. Multiple Query windows may be open at one time. The Query window (Figure 3-3) has a black title bar above the window, containing the text "Query File Name: <queryname> Schema: <schemaname>." Query windows have a menu bar containing the menus "Query" (Figure 3-4) and "Edit" (Figure 3-5).

Table 3-2. Query Window Menus

Menu	Menu Items	Result of Selection
Query	Save	<saves open query>
	Save As...	Save As dialog
	Print	<prints current query>
	Delete...	Delete? message
	Close Query...	Save? message
	Execute	<executes query and opens text display containing result of query>
Edit	Clear	<clear text in selected clause box>
	Find Node...	Find Node dialog

The Query window contains three edit boxes, stacked vertically, which the user may select for typing in a query clause. The three edit boxes are entitled "Type a Select clause:," "Type a From clause:," and "Type a Where clause:." When the user clicks the mouse button when the cursor is over one of these edit boxes, the selected box receives a double border, and the user may type into it. The first word of the clause, corresponding to the name of the clause, is already present in the edit box when selected.

3.4.4 Dialog Boxes

There are a several standard dialog boxes in the ANSWER interface. Table 3-3 lists the individual dialog boxes, how each is invoked, and the contents of each.

Dialog boxes have a black title bar containing the dialog title, and a window body containing the interaction objects to support the dialog (textual lists, edit boxes, radio buttons, and pushbuttons) and pushbuttons to close the dialog (variants of "OK" and "Cancel" commands). Dialog boxes appear "on top" of all other windows, and are application modal; that is, the user must select one of the "close dialog" pushbuttons, dismissing the dialog box, before being allowed to further interact with other ANSWER components. Only one dialog box may be present at any time.

Figures 3-6 through 3-14 show the dialog boxes.

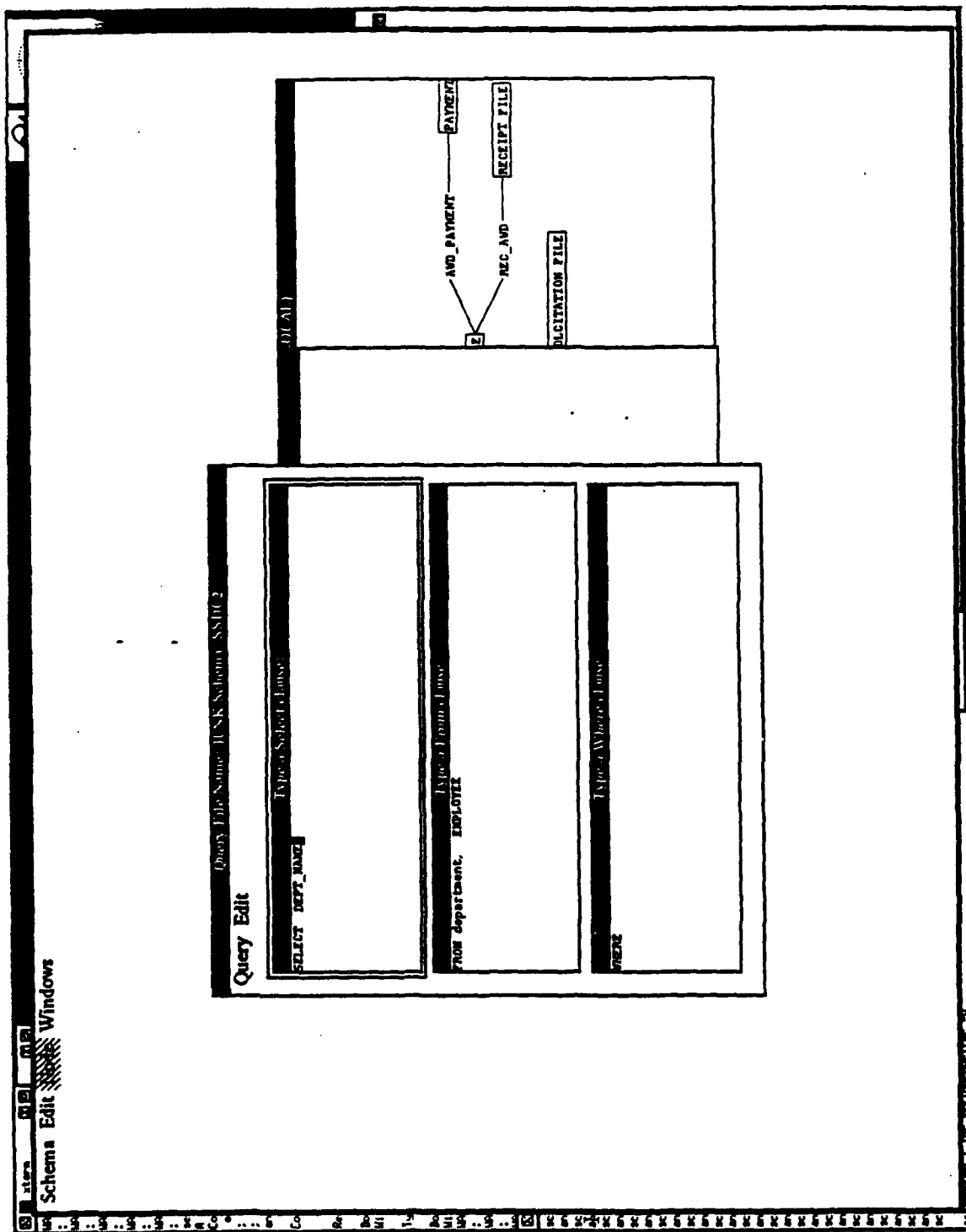


Figure 3-3. Query Window

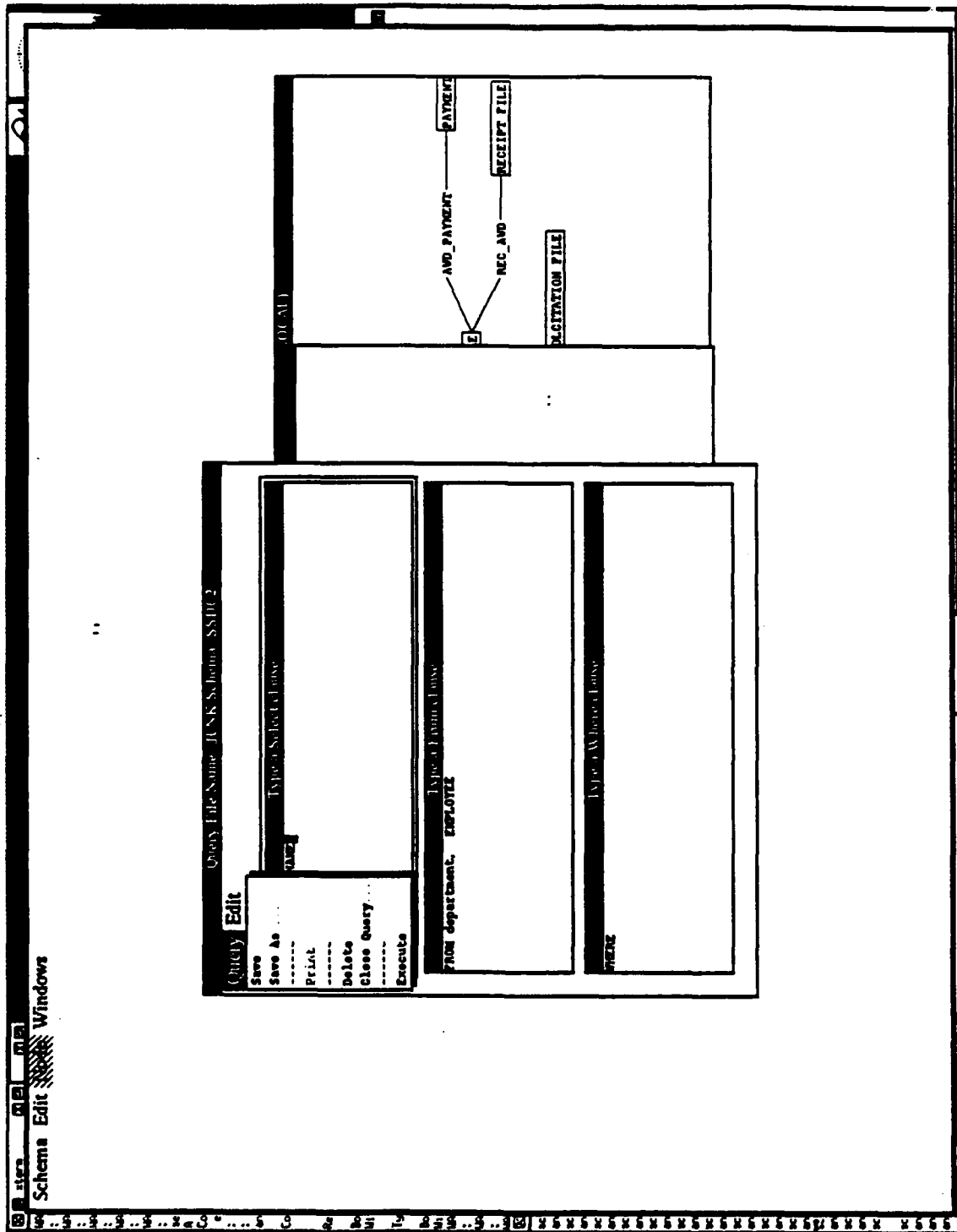


Figure 3-4. Query Window: Query Menu

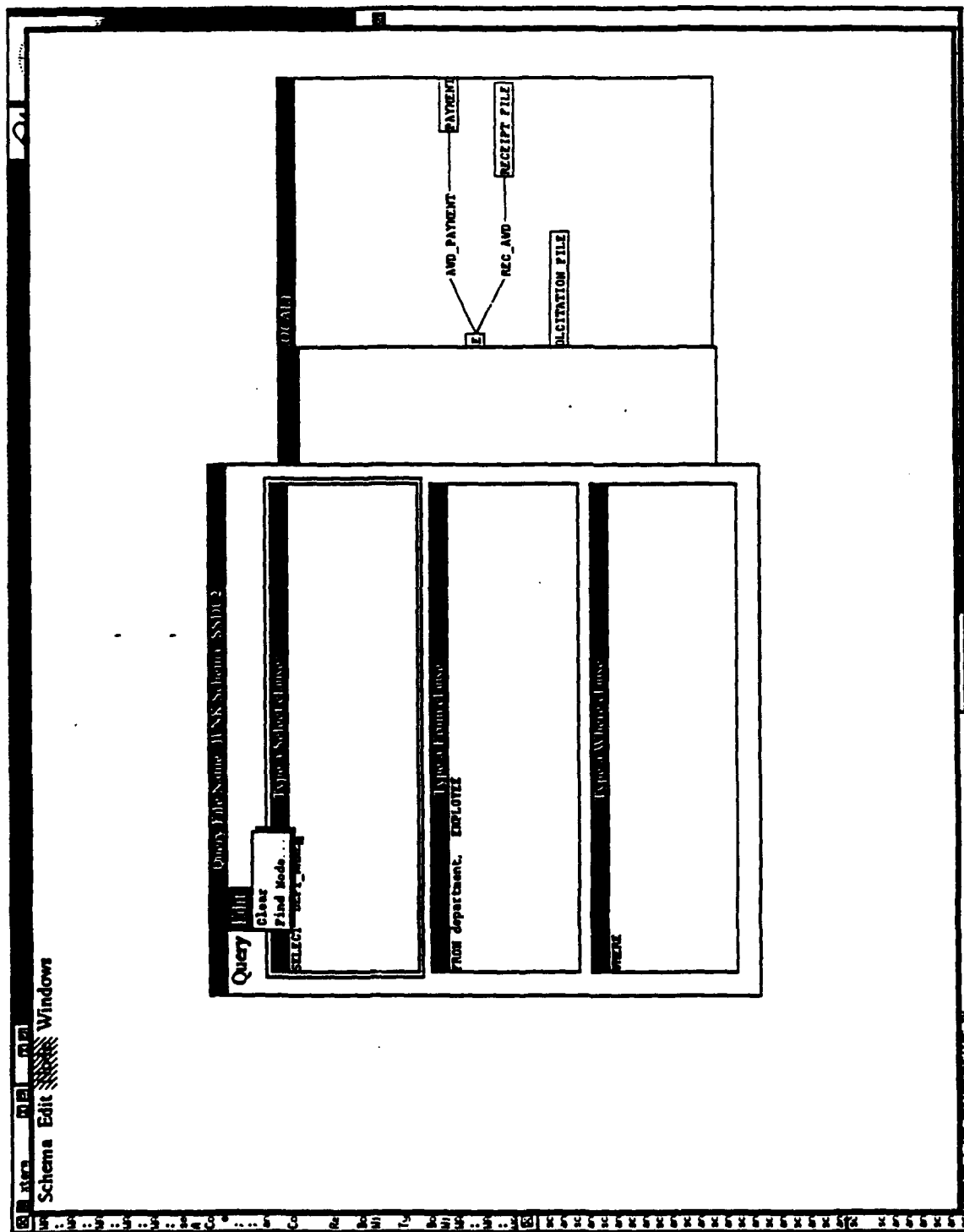


Figure 3-5. Query Window: Edit Menu

Table 3-3. Standard ANSWER Dialog Boxes

Dialog Box Name	Access Method	Components
Open Schema	Startup Schema→ Open Schema... Edit→ Integrate...	<ul style="list-style-type: none"> • Textual list of available schemas, entitled "Schemas"; selected schema is highlighted • Text box showing selected schema • Radio buttons (IRDS, Local, Relational) to select schema category (selected button determines contents of textual list); disabled if called by "Integrate" menu item • Pushbutton (Open) to open the selected schema; disabled until a schema has been selected from the textual list; if integration is occurring, opens the named schema as the second schema for integration and opens the "Assert Equivalence" dialog box • Pushbutton (Cancel) to cancel the dialog without opening a schema (also canceling integration if called by "Integrate" menu item)
Save As	Schema→ Save Schema As... Query→ Save As...	<ul style="list-style-type: none"> • Dialog box title: "Save Schema" or "Save Query" • Textual list of pre-existing schemas or queries, entitled "Schemas" or "Queries"; all schema names are disabled, and cannot be selected by the user • Edit box entitled "Type Schema Name:" or "Type Query Name:" to allow the user to enter the desired name for the schema or query • Radio buttons (IRDS, Local) to select schema category; disabled for query • Pushbuttons (Save, Cancel) to save the schema or query under the entered name, or cancel the dialog without saving. The "Save" pushbutton is disabled until a schema or query name has been entered.
Add Node	Schema→ Add Node... Schema→ New Schema...	<ul style="list-style-type: none"> • Dialog box title: "Add New Node" • Edit box for user to type name of new node, entitled "Type Node Name:" • Radio buttons (Category, Entity) to specify type for new node • Pushbutton "OK" to close the dialog and add new node to active schema or open a new schema ("Untitled") with named node. • Pushbutton "Cancel" to close the dialog and make no changes

Table 3-3. Standard ANSWER Dialog Boxes (continued)

Find Node	Node→ Find Node...	<ul style="list-style-type: none"> • Dialog box title: "Find Node" • Textual list of nodes or attributes in the active schema, entitled "Nodes" or "Attributes," selectable by the user (contents determined by radio buttons "Attributes," "Nodes") • Edit box entitled "Search For:," user may enter a text string • Pushbutton "Search" to allow to search the node list for a string (contained in the "Search For:" edit box), and display only those nodes containing that string • Radio buttons (Attributes, Nodes) to specify the contents of the textual list; disabled if called from a Query window • Pushbutton "Find" to find the selected node in the textual list in the current schema; disabled if no node is selected from the textual list; closes the dialog and highlights the found node • Pushbutton "Cancel" to close the dialog without locating a node in the schema
Open Query	Edit→ Open Query...	<ul style="list-style-type: none"> • Dialog box title: "Open Query" • Textual list of existing queries for the active schema, entitled "Queries," selectable by the user • Text box showing selected query from textual list • Pushbutton "Open" to close the dialog and open the named query, enabled if a query is picked in the textual list and named in the text box • Pushbutton "Cancel" to close the dialog and open no query
Edit Attribute	Node→ Edit Attribute...	<ul style="list-style-type: none"> • Dialog box title: "Edit Node (Node: <node name> Type: <type>)" • Edit box for user to type name of new attribute, entitled "Type Attribute Name:" • Textual list of attributes of the selected node, entitled "Attributes," selectable by the user • Pushbutton "Add" to allow the user to add a new attribute, enabled if user has typed into the "Type Attribute Name:" edit box • Radio buttons (Relational, String, Integer, Alphanumeric) to specify the type of the new attribute • Radio buttons (Key, Non-Key) to specify whether the attribute is key or non-key • Pushbutton "OK" to close the dialog and accept changes to the selected node. • Pushbutton "Cancel" to close the dialog and make no changes to the selected node

Table 3-3. Standard ANSWER Dialog Boxes (continued)

Edit Node	Node→ Edit Node...	<ul style="list-style-type: none"> • Dialog box title: "Edit Node (Node: <node name> Type: <type>)" • Textual list of nodes related to the node that is being edited, entitled "Nodes,"selectable by the user • Pushbutton "Add" to allow the user to add a new relationship (disabled) • Radio buttons (Contained In, Contains) to specify the type of new relationship (disabled) • Pushbutton "Specify Relation.." to allow the user to enter a relationship type; disabled until a node is picked from the "Node" textual list. • Textual list of currently specified relationships, entitled "Relationships," selectable by the user • Two sets of radio buttons (One, Many) specify the nature of the relationship between the two nodes • Pushbutton "Delete" to allow the user to delete a selected relationship; disabled if no relationship is picked in the "Relationships" textual list • Pushbutton "OK" to close the dialog and accept changes to the edited node.
Assert Attribute Equivalence	Open Schema dialog during integration	<ul style="list-style-type: none"> • Dialog box title: "Assert Attribute Equivalence (<schema name1> & <schema name2>)" • Textual list of attributes and entities for the active schema, entitled "<schema name1> (Attr-Ent)," selectable by the user; after an item has been asserted as equal to another item, it becomes grayed in the list and is not selectable by the user • Textual list of attributes and entities for the second schema, entitled "<schema name2> (Attr-Ent)," selectable by the user; after an item has been asserted as equal to another item, it becomes grayed in the list and is not selectable by the user • Textual list of attributes and entities that have been asserted as equivalent for the two schemas, entitled "Equivalent Attributes," selectable by the user • Pushbutton "Equal" to allow equivalence assertion, enabled if an item is selected from both of the textual lists from the schemas; asserted items appear in textual list of equivalent attributes • Pushbutton "Delete" to delete an equivalence pair; enabled if item selected from the list of declared equivalences • Pushbutton "Continue" to continue the integration; enabled if at least one set of items have been declared equivalent; opens Assert E/C dialog • Pushbutton "Cancel" to close the dialog and cancel the integration

Table 3-3. Standard ANSWER Dialog Boxes (Concluded)

Assert E/C	Assert Equivalence dialog during integration	<ul style="list-style-type: none"> • Dialog box title: "Assert E/C Equivalence (<schema name1> & <schema name2>)" • Textual list of unasserted pairs of entities and categories in the two schemas, entitled "Unasserted E/C Pairs," selectable by the user; after an item has been asserted, it becomes grayed in the list and is not selectable by the user • Radio buttons (Equal, Contains→, ←Contained In, Not Equal, Overlap, Unknown) to allow the user to specify the type of relationship between the named nodes • Pushbutton "Assert" to allow E/C equivalence assertion, enabled if an item is selected from the textual lists of unasserted pairs; asserted pair appears in textual list of asserted pairs • Textual list of Asserted E/C pairs, selectable by the user • Pushbutton "Delete" to delete an equivalence pair; enabled if item selected from the list of declared equivalences • Pushbutton "Integrate" to command the integration to take place; enabled if at least one set of items have been declared equivalent • Pushbutton "Cancel" to close the dialog and cancel the integration
------------	--	--

3.4.5 Message Boxes

Table 3-4 lists the standard message boxes in the ANSWER interface. Message boxes have a double border, with the inside border thicker than the outside border, text stating the system question (which may be in the form of a statement that the user must confirm or reject), and two pushbuttons representing user responses to the system question. Message boxes appear "on top" of all other windows, and are application modal; that is, the user must select one of the message box pushbuttons, thus dismissing the message box, before being allowed to further interact with other ANSWER components. Only one message box may be present at any time.

Figures 3-15 through 3-18 show the message boxes.

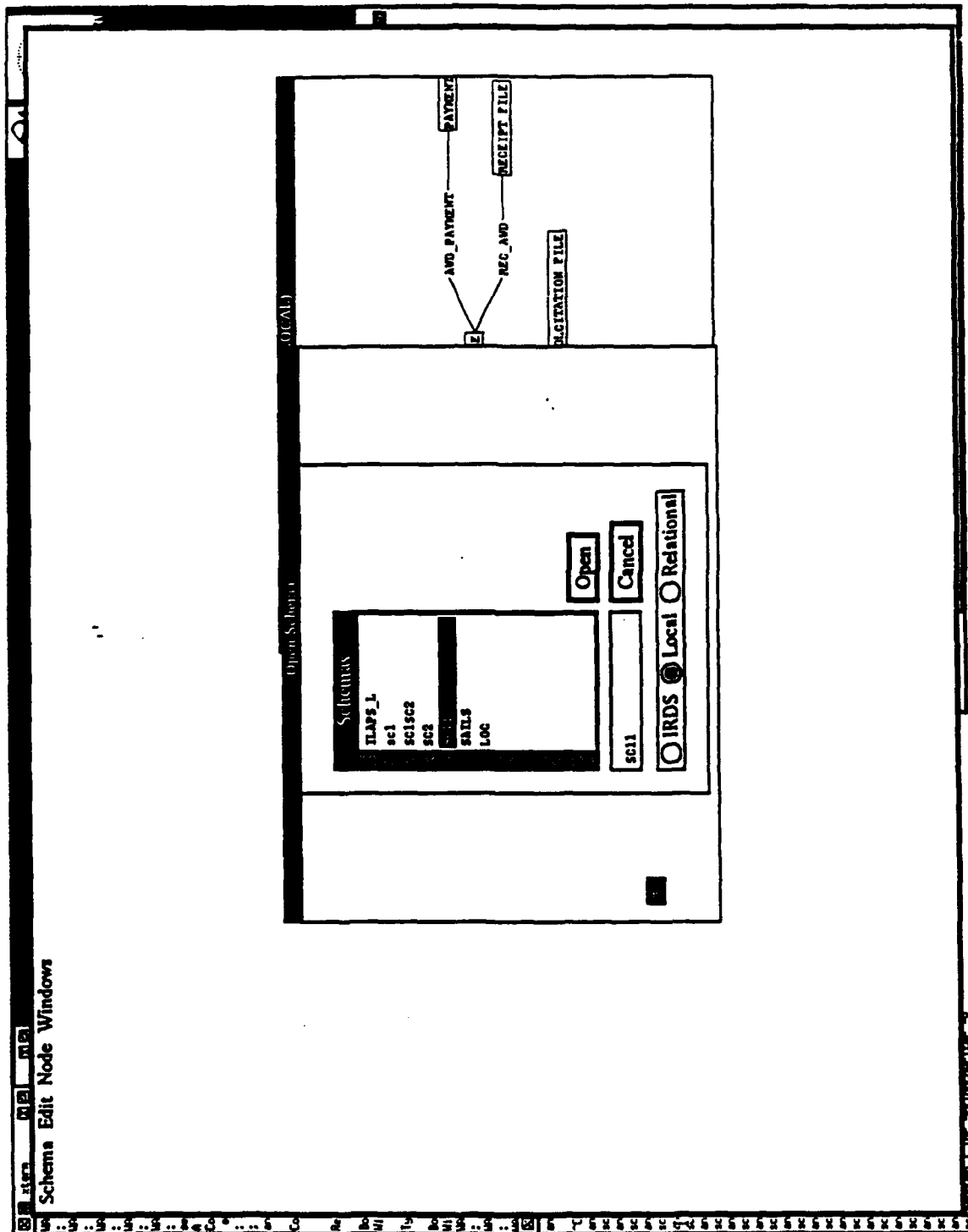


Figure 3-6. Open Schema Dialog Box

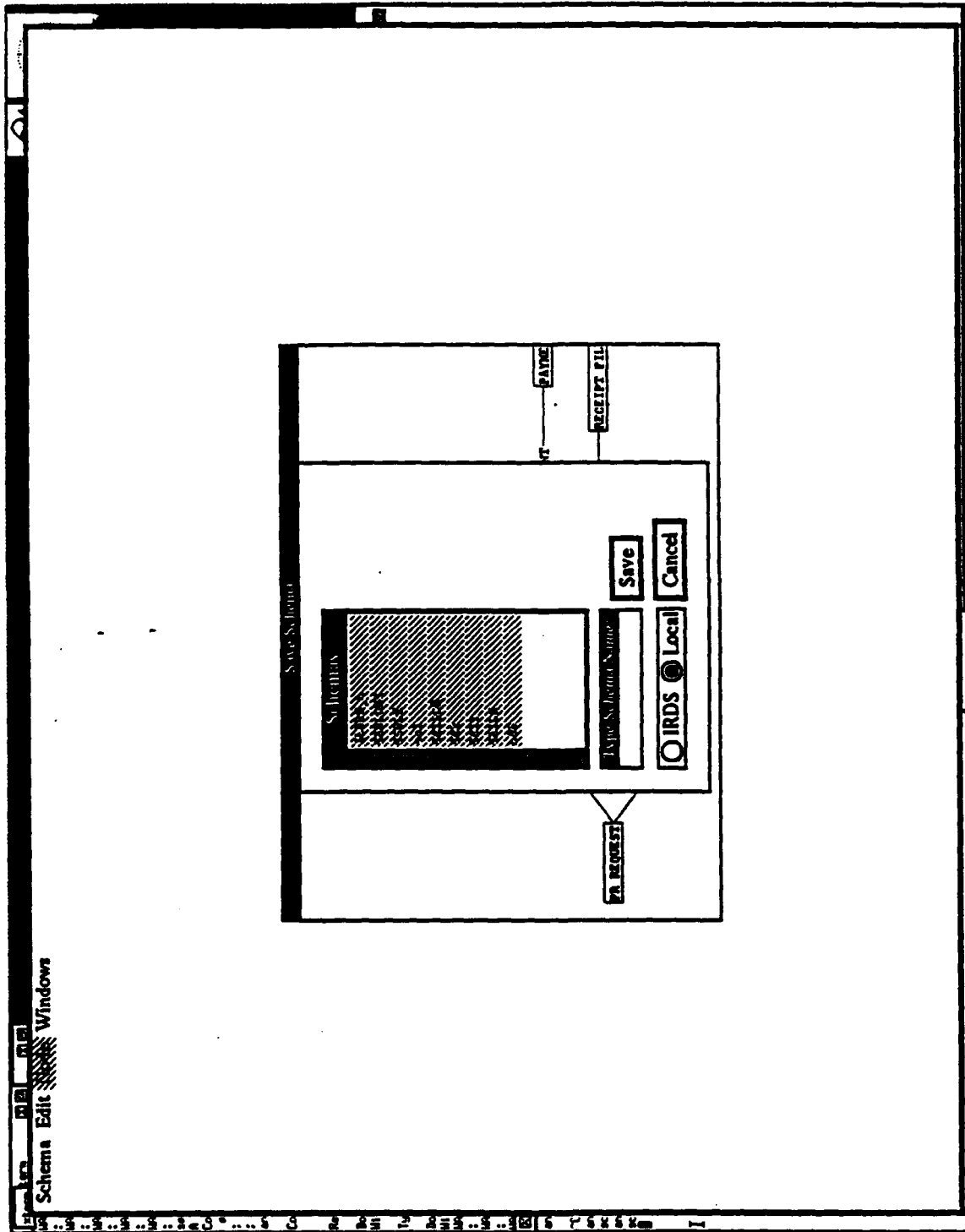


Figure 3-7. Save As Dialog Box

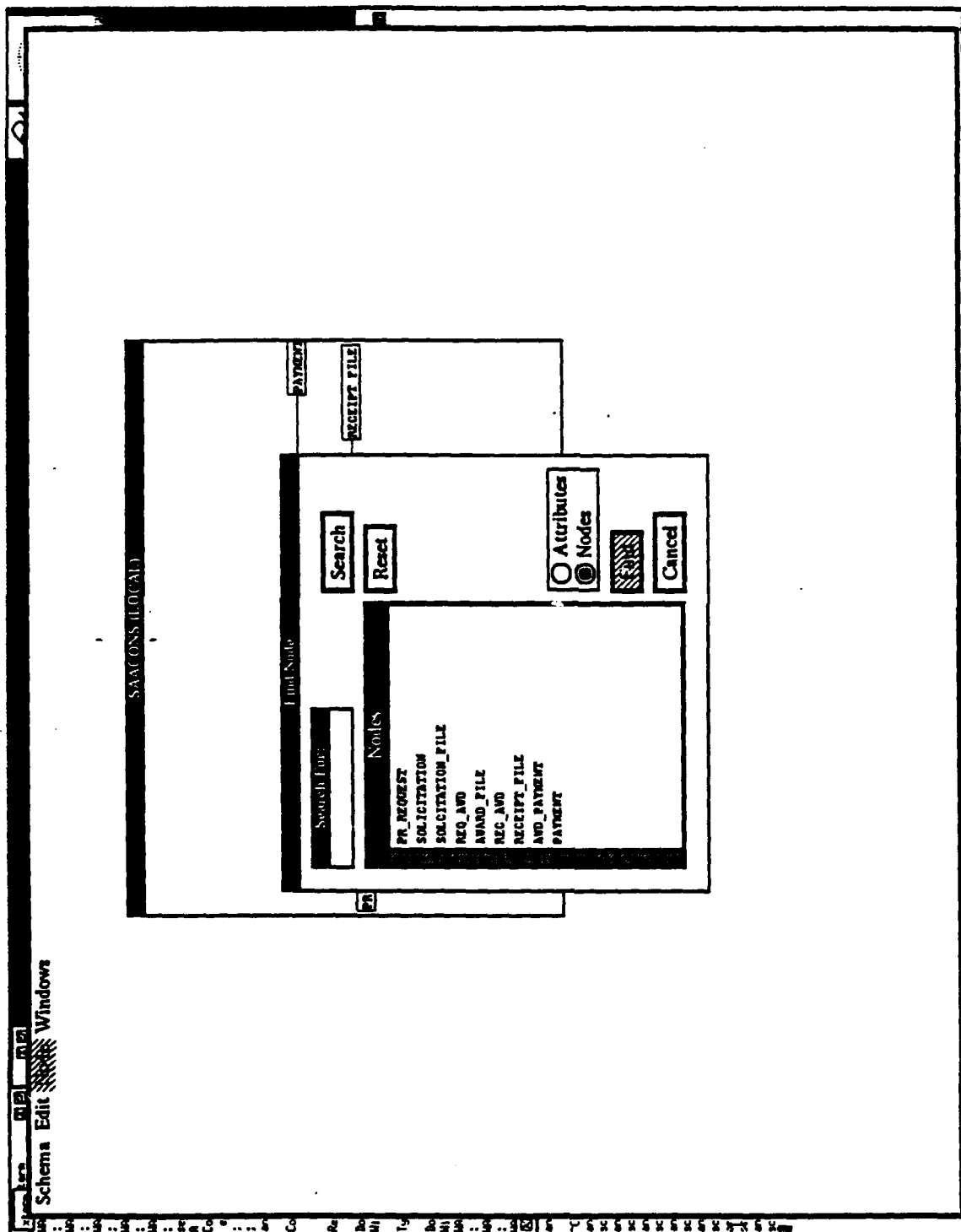


Figure 3-9. Find Node Dialog Box

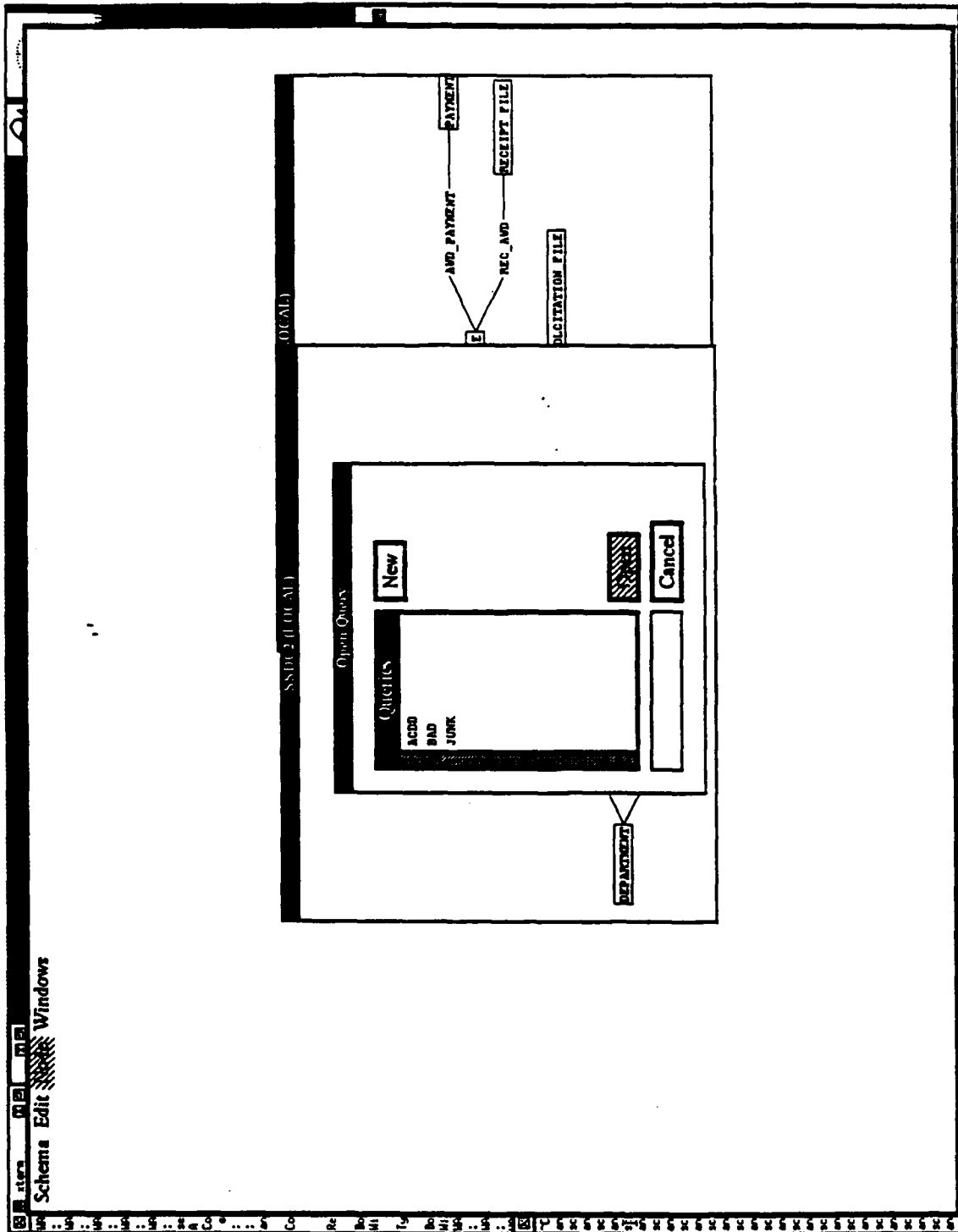


Figure 3-10. Open Query Dialog Box

Schema Edit Node Windows

File Edit View Help

Entity: SAACONS (LOCAL)

Type Attribute Name:

Attributes:

- AUTH_DATE
- AVD_DELORD
- AVD_FSC_CLASS
- AVD_P11M
- AVD_STATUS
- DELTO
- PR_RUN
- P_ID

Delete

☐ Rational
☐ String
☐ Integer
☒ Alphanumeric

☒ Key
☐ Non-Key

OK Cancel

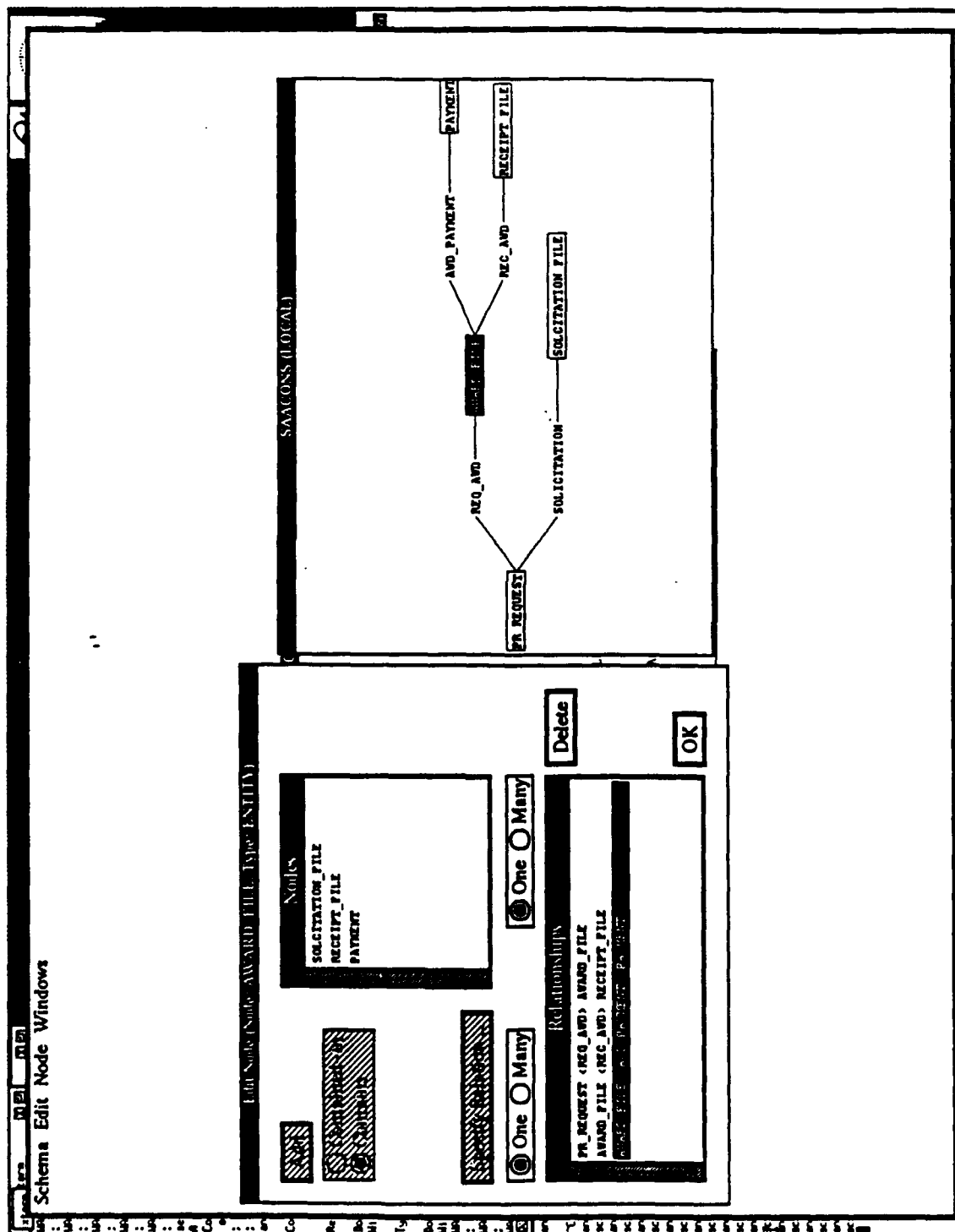
Entity: SAACONS (LOCAL)

Diagram:

```

    graph LR
      PR_REQUEST[PR REQUEST] --> REQ_AND[REQ AND]
      PR_REQUEST --> SOLICITATION[SOLICITATION]
      REQ_AND --> REQ_FILE[REQ FILE]
      REQ_FILE --> AVD_PAYMENT[AVD_PAYMENT]
      REQ_FILE --> REC_AND[REC AND]
      REC_AND --> REC_FILE[RECEIPT FILE]
      SOLICITATION --> SOLICITATION_FILE[SOLICITATION FILE]
  
```

Figure 3-11. Edit Attribute Dialog Box



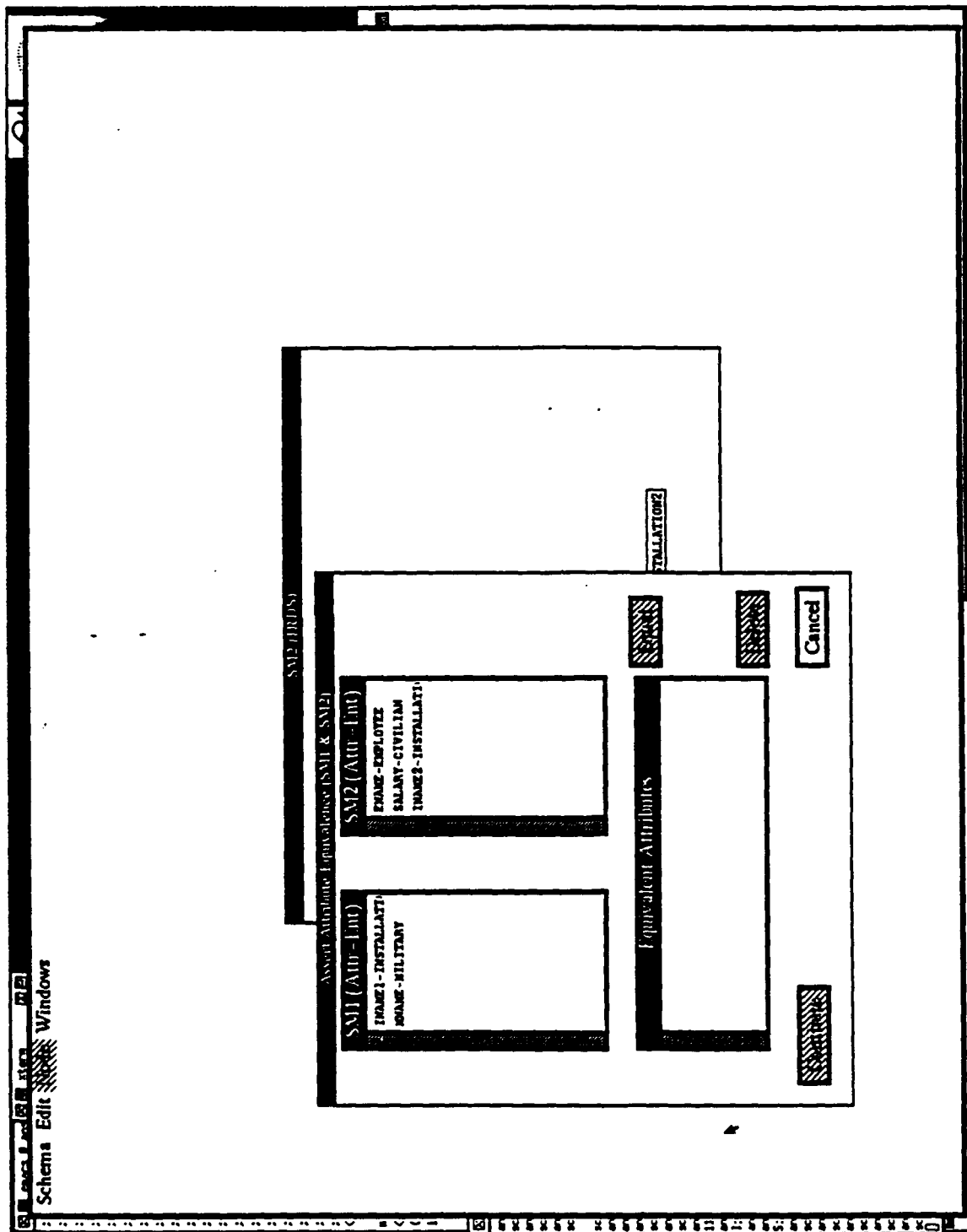


Figure 3-13. Assert Attribute Equivalence Dialog Box

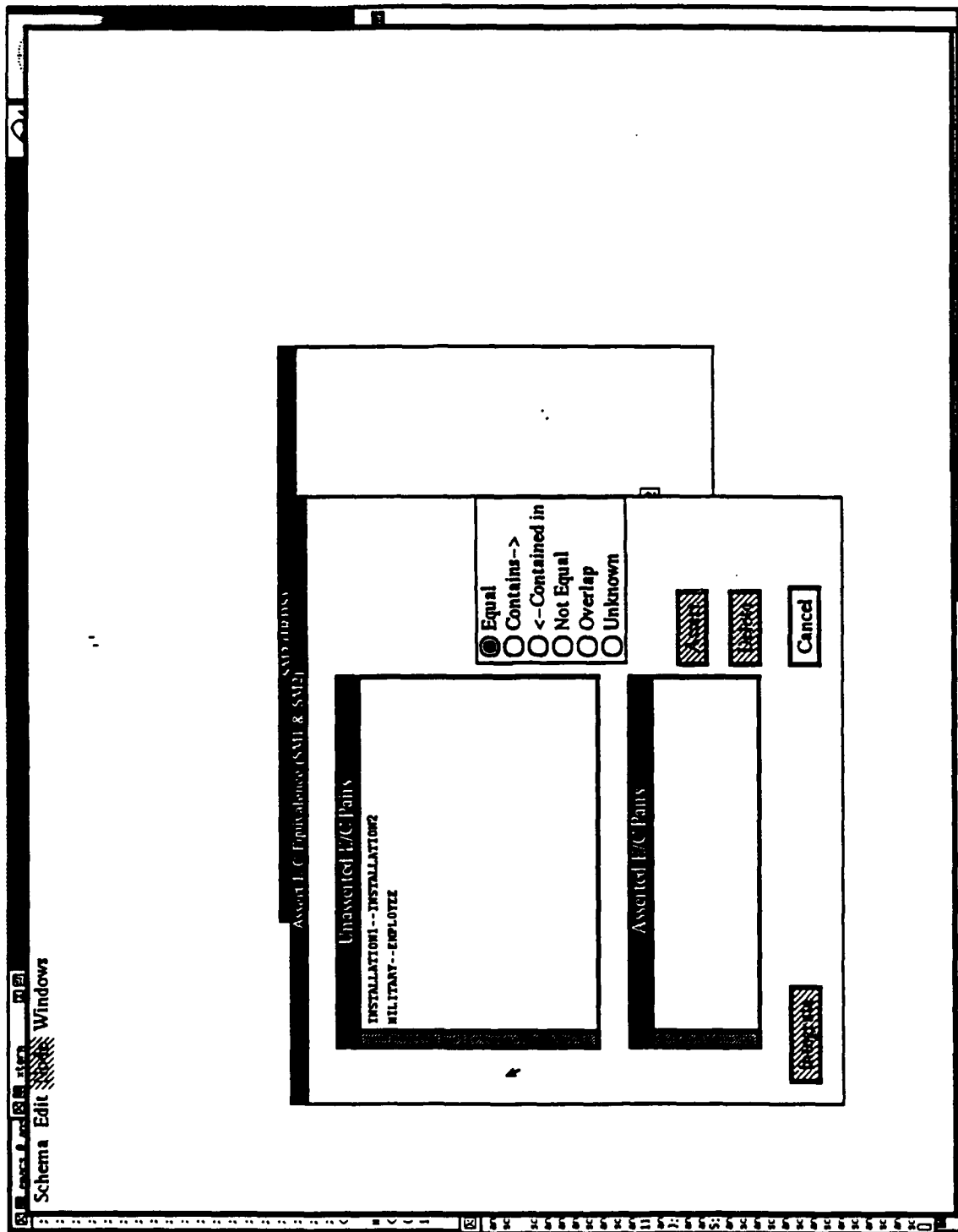


Figure 3-14. Assert E/C Equivalence Dialog Box

Table 3-4. Standard ANSWER Message Boxes

Message Box Name	Access Method	Components
Delete?	Schema → Delete Schema... Node → Delete Me... Query → Delete...	<ul style="list-style-type: none"> • System text: "Do you want to delete <name>?" • Pushbuttons (Yes, No) to accept or cancel the deletion action.
Quit?	Schema → Quit...	<ul style="list-style-type: none"> • System text: "Please confirm to exit ANSWER." • Pushbuttons (Yes, No) to accept or cancel the application exit command.
Legend	Edit → Show Legend...	<ul style="list-style-type: none"> • System text: "The node types and their depictions are as follows: " • Graphics resembling node buttons, labeled with each node type • Pushbutton (OK) to close the legend
Save?	Query → Close Query...	<ul style="list-style-type: none"> • System text: "Do you want to save <queryname>?" • Pushbuttons (Yes, No) to accept or cancel the save command

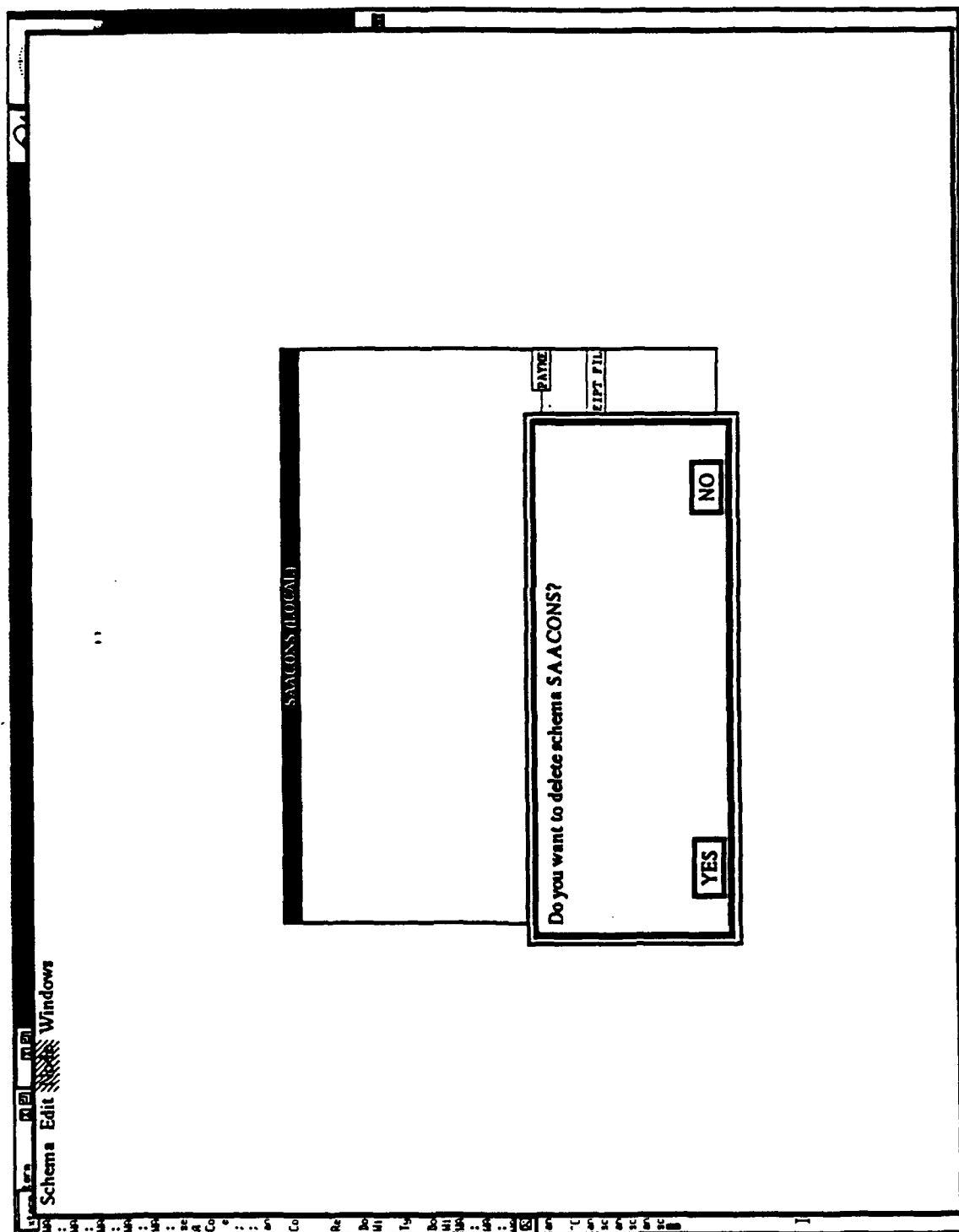


Figure 3-15. Delete? Message Box

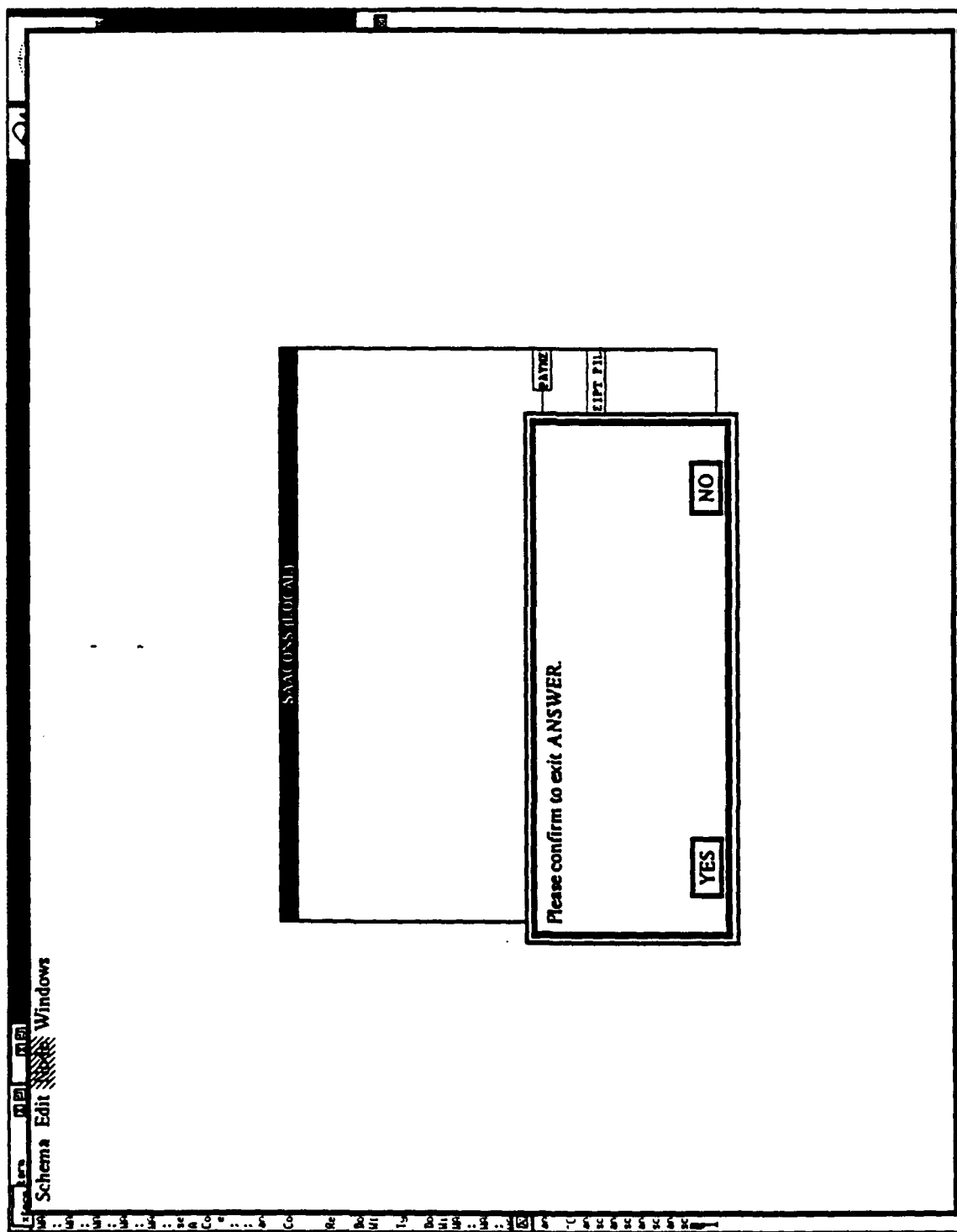


Figure 3-16. Quit? Message Box

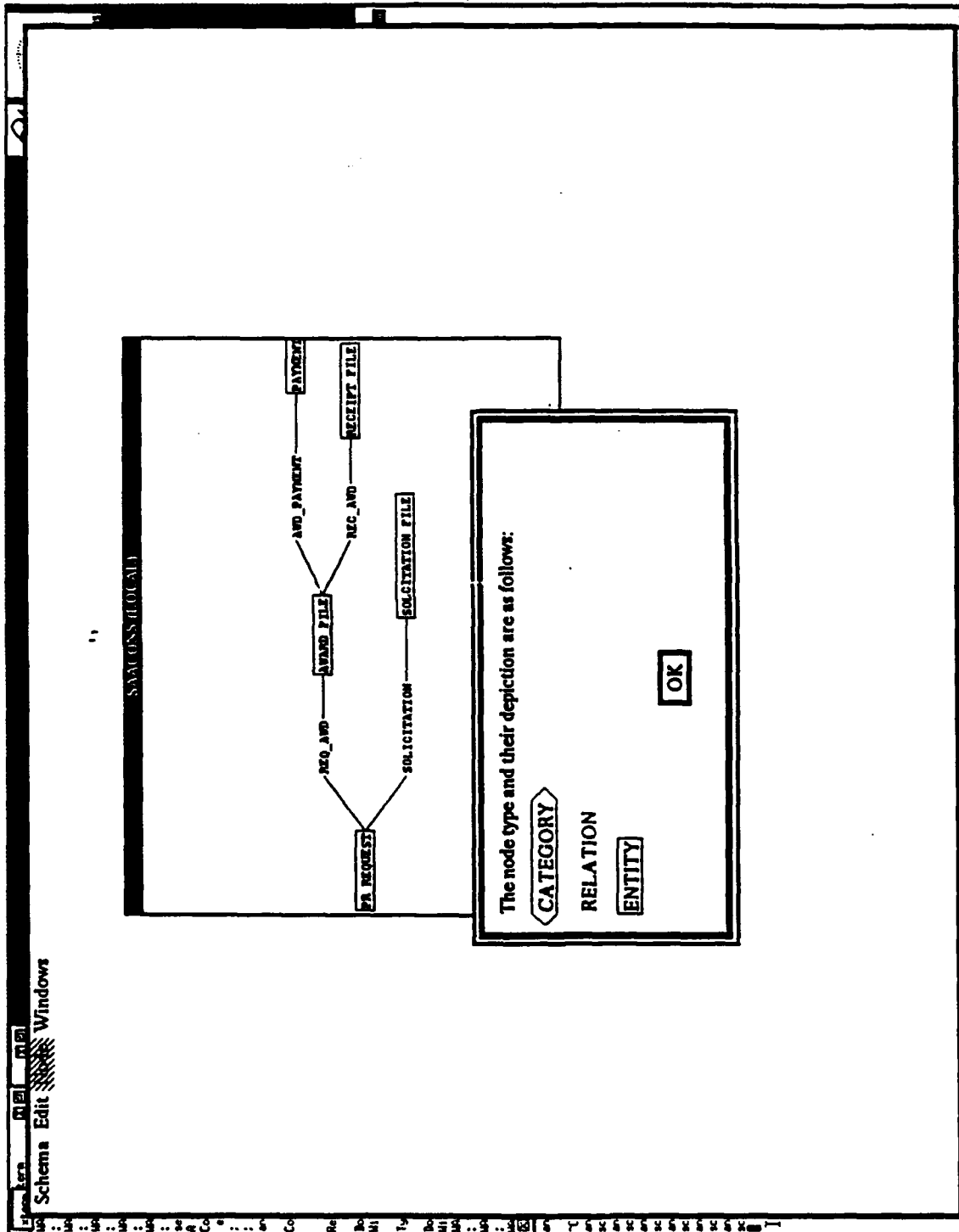


Figure 3-17. Legend Message Box

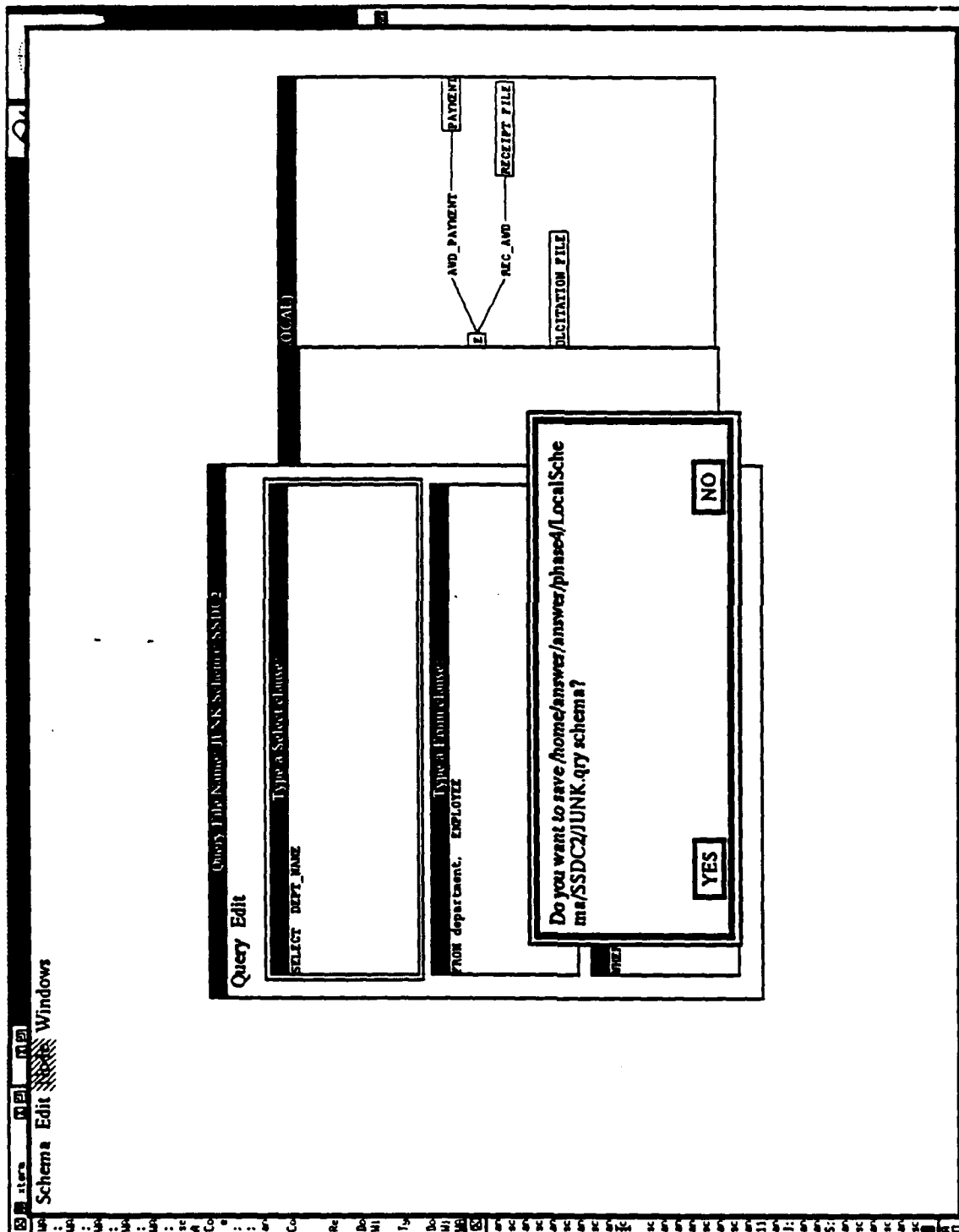


Figure 3-18. Save? Message Box

Section 4

ANSWER Installation and Startup

This section describes how to install and start up the ANSWER system on a single Sun 3. ANSWER can also run on as many as three Sun 3 systems.

4.1 ANSWER System Installation

The ANSWER installation tapes assume a single Sun 3 with at least 12 megabytes of memory, a 68881 floating point coprocessor, a 300-megabyte disk and SunOS 4.0.3. The installation occupies the complete 300 megabyte disk, so any previous contents will be lost. The installation includes the following commercial software packages:

- Oracle DBMS version 6.0, including Pro*C, SQL*Plus, and SQL*Net;
- Informix DBMS version 4.00, including Informix Embedded SQL for C and Informix-Net (I-NET);
- Allegro Common LISP (CL) release 3.1.13 and Common Windows on X (xcw), release 1.3.final.11.2.

The user must hold licenses for these packages prior to installation. ANSWER also incorporates X11 R4 and the twm window manager, which are public domain software.

Installation involves the following steps:

- Format the Sun's disk, following the instructions in the Sun System Administration manual. Create a disk label with the following data:
 - a partition—8 meg,
 - b partition—50 meg,
 - g partition—41 meg,
 - h partition—205 meg.
- Boot Sun Operating System version 4.0.3 from the Sun distribution tape.
- Type the following commands:

```
mkdir /sda
mkdir /sdg
mkdir /sdh
```
- Mount the "a" partition on /sda.

- Insert the tape marked “/dev/sd0a” in the tape drive.
- Change working directory to /sda and type:

```
restore -rf /dev/rst0
```

The restore process should take about five minutes.

- Mount each partition on a different directory (e.g., /dev/sd0g on /sdg, /dev/sd0h on /sdh) by changing the working directory to each directory, mounting the appropriate tape and typing the restore command exactly as above. The system will prompt you for the next tape in a multitape restore. There is no tape for the “b” partition (swap and page space). The “g” partition should take about 45 minutes. The “h” partition should take one to two hours.
- Change working directory to /usr/kvm/mdec and type:

```
installboot -vlt /sda/boot bootsd /dev/rsd0a
```

- Reboot the machine.

4.2 Starting up Informix

When the machine is booted, commands within the /etc/rc.local file start Informix automatically. No user interaction is required.

4.3 Starting up Oracle

The user must do the following to start Oracle following system reboot:

- Log in as user “oracle.” No password is required.
- Type the command “rm dbs/sgadefanswer.dbf” to delete the file “dbs/sgadefanswer.dbf” if it exists. It may be present if Oracle was not shut down prior to the last system shutdown or reboot.
- Use the Oracle “sqldba” command to start Oracle. Executing this command will result in the following typeout:

```
SQL*DBA: Version 6.0.26.8.2 - Production on Thu May 30 11:10:12 1991
Copyright (c) Oracle Corporation 1979, 1988. All rights reserved.
ORACLE RDBMS V6.0.26.8.2, transaction processing option - Production
SQLDBA>
```


- At the "SQLDBA>" prompt, type "startup pfile=dbs/initanswer.ora."

This should produce the following results:

```
ORACLE instance started
Database mounted
Database opened
Total System Global Area 9044160 bytes
Fixed Size 17828 bytes
Variable Size 801564 bytes
Database Buffers 8192000 bytes
Redo Buffers 32768 bytes
SQLDBA>
```

- At the "SQLDBA>" prompt, type "exit" to quit sqldb.
- Log out as user "oracle."

Further information about the sqldb program can be found in the *Oracle RDBMS Database Administrator's Guide*.

To shut Oracle down prior to a system shutdown or reboot, use sqldb again and type:

```
shutdown normal
```

at the SQLDBA> prompt.

4.4 Starting and Exiting ANSWER

To start the ANSWER system, perform the following steps:

- Log in as user "answer." No password is required.
- Type the command "startx" to start X-Windows and create a standard set of windows.
- After the windows appear, place the mouse cursor in the upper right window (running the UNIX shell) so that user type-in is directed at the shell. Type "answer" to start the ANSWER system.
- ANSWER will ask you whether you want to connect to IRDS. Type "yes" or "no" in reply. Connecting to IRDS is a prerequisite for schema integration, but it takes a few minutes.
- After a sequence of messages regarding establishment of connections, the ANSWER main window should appear. Section 3 describes how to interact with ANSWER.

Do the following to exit ANSWER and log out:

- Quit ANSWER using the Quit item in the Schema menu. The ANSWER main window should disappear.
- Position the mouse cursor on the gray background outside of any window. Depress the right mouse button, move the mouse cursor to select the Exit X11 menu item and release the right mouse button. All windows should disappear and a UNIX shell prompt should appear.
- Log out if desired.

Section 5

References

- [ANSWER88] Army's Nonprogramming System for Working Encyclopedia Requests (ANSWER)—Interim Technical Report, Honeywell Inc., December 1988.
- [ANSWER89a] Army's Nonprogramming System for Working Encyclopedia Requests (ANSWER)—Final Report, Honeywell Inc., June 1989.
- [ANSWER89b] ANSWER Phase II Final Report, Honeywell Inc., December 1989.
- [ANSWER90] ANSWER Phase IIIA Final Report, Honeywell Inc., December 1990.
- [ANSWER91] ANSWER Phase IIIB Final Report, Honeywell Inc., March 1991
- [AR 25-9] Army regulation 25-9, Army Data Management and Standards Program, 1989.
- [Chen76] P. Chen, "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Transactions on Database Systems*, Vol. 1, March 1976.
- [Elma80] R. Elmasri and Gio Wiederhold, "Properties of Relationships and Their Representation," *Proc. National Computer Conference*, 1980.
- [Elma85] R. Elmasri, J. Weeldreyer, and A. Hevner, "The Category Concept: An Extension to the Entity-Relationship Model," *Data and Knowledge Engineering*, No. 1, 1985.
- [ISO/IEC 9075] "Database Language—SQL with Integrity Enhancement," International Organization for Standardization, 1989.
- [Bati92] C. Batini, S. Ceri, and S. B. Navathe, *Conceptual Database Design—An Entity-Relationship Approach*, Benjamin/Cummings, 1992.
- [NBSIR88-3700] A. Goldfine and P. Konig, "A Technical Overview of the Information Resource Dictionary System (2nd Edition)," U.S. Department of Commerce, National Bureau of Standards, January 1988.